9.0

Vývoj aplikací pro produkt IBM MQ

Poznámka

Než začnete používat tyto informace a produkt, který podporují, přečtěte si informace, které uvádí “Poznámky” na stránce 1323.

Toto vydání se vztahuje k verzi 9 vydání 0 produktu IBM® MQ a ke všem následujícím vydáním a modifikacím, dokud nebude v nových vydáních uvedeno jinak.

Když odešlete informace do IBM, udělíte společnosti IBM nevýlučné právo použít nebo distribuovat informace libovolným způsobem, který společnost považuje za odpovídající, bez vzniku jakýchkoliv závazků vůči vám.

Obsah

Vyvíjení aplikací pro IBM MQ

Můžete vyvíjet aplikace k odesílání a přijímání zpráv a ke správě správců front a souvisejících prostředků. IBM MQ podporuje aplikace napsané v mnoha různých jazycích a rámcích.

Chcete-li se dozvědět více o vývoji aplikací pro produkt IBM MQ, navštivte produkt IBM Developer:

- LearnMQ *(naučte se základy, spusťte demoverzi, kódujte aplikaci a získejte další rozšířené výukové programy)*
- Soubory ke stažení MQ pro vývojáře *(včetně vydání Developer zdarma a zkušebních verzí)*

Pokud jste obeznámeni s koncepcemi popsanými v následujících sekcích, může být také snazší vyvíjet aplikace.

- “Koncepty vývoje aplikací” na stránce 6
- “Aspekty návrhu pro aplikace produktu IBM MQ” na stránce 43

Podpora pro objektově orientované jazyky a rámce

Produkt IBM MQ poskytuje základní podporu pro aplikace vyvinuté v následujících jazycích a rámci:

- JMS
- Java
- Jazyk C++
- .NET
- ActiveX (zamítnutý; použijte .NET)

Další informace najdete v tématu “Objektově orientované aplikace” na stránce 10.

Produkt .NET podporuje aplikace vyvinuté v mnoha jazycích. Pro ilustraci použití tříd produktu IBM MQ for .NET pro přístup k frontám produktu IBM MQ obsahuje dokumentace k produktu MQ informace pro následující jazyky:

- C# (příklad kódu)
- JAZYK C++
- Visual Basic

Viz “Zápis a implementace programů produktu IBM MQ .NET” na stránce 529.

ULW Produkt IBM MQ také podporuje rozhraní API produktu MQ Light , které implementuje protokol OASIS AMQP 1.0 . Pro následující jazyky jsou k dispozici rozhraní API systému zpráv:

- Node.js
- Ruby
- Java
- Python
- Maven (projekt kostry; používá rozhraní API Java)
- Gradle (projekt kostry; používá rozhraní API Java)

Další informace najdete v tématu “Vyvíjení klientských aplikací AMQP” na stránce 654.

Následující jazykové vazby jsou poskytovány tak, jak jsou:

- Vazba Go
- implementace rozhraní JavaScript API, která pracuje s aplikacemi Node.js

Podpora pro programová rozhraní REST API

Produkt IBM MQ poskytuje podporu pro následující programová rozhraní REST API pro odesílání a příjem zpráv:

- V 9.0.4 IBM MQ messaging REST API
- z/OS IBM z/OS Connect EE
- IBM Integration Bus
- BránaIBM DataPower Gateway

Viz “Vývoj aplikací REST s produktem IBM MQ” na stránce 669, a také výukový program Začněte pracovat se systémem zpráv REST API produktu IBM MQ v oblasti IBM MQ vývojáře IBM . Tento výukový program obsahuje příklady v následujících jazycích za předpokladu, že jsou použity s produktem IBM MQ messaging REST API:

- Příklad použití rozhraní REST API systému zpráv MQ
- Příklad Node.js s použitím modulu HTTPS
- Příklad Node.js s modulem Promise

Podpora procedurálních programovacích jazyků

Produkt IBM MQ poskytuje podporu pro aplikace vyvinuté v následujících procedurálních programovacích jazycích:

- P
- Windows Visual Basic (pouze systémyWindows)
- COBOL
- z/OS Assembler (pouzeIBM MQ for z/OS)
- IBM i RPG (pouzeIBM MQ for IBM i)
- z/OS PL/I (pouzeIBM MQ for z/OS)

Tyto jazyky používají rozhraní fronty zpráv (MQI) pro přístup ke službám front zpráv. Viz “Vyvíjení aplikací MQI s produktem IBM MQ” na stránce 685. Všimněte si, že objektový model produktu IBM MQ používaný v jazycích a rámcích orientovaný na objekty poskytuje další funkce, které nejsou k dispozici pro procedurální jazyky pomocí rozhraní MQI.

Související pojmy

“Vývoj aplikací Microsoft Windows Communication Foundation (WCF) s IBM MQ” na stránce 1217
Vlastní kanál Microsoft Windows Communication Foundation (WCF) pro produkt IBM MQ odesílá a přijímá zprávy mezi klienty WCF a službami.

Související úlohy

“Vyvíjení aplikací pro MQ Telemetry” na stránce 1194

Související informace

IBM Message Service Client for .NET

Koncepty vývoje aplikací

K zápisu aplikací IBM MQ můžete použít volbu procedurálních nebo objektově orientovaných jazyků. Odkazy v tomto tématu použijte pro informace o koncepcích produktu IBM MQ , které jsou užitečné pro vývojáře aplikací.

Než začnete navrhovat a psát své aplikace produktu IBM MQ , seznamte se se základními koncepcemi produktu IBM MQ , viz témata v tématu Technický přehled. Informace o typech aplikací, které můžete zapsat pro IBM MQ, viz “Vyvíjení aplikací pro IBM MQ” na stránce 5.

Pomocí následujících odkazů můžete zjistit informace o koncepcích produktu IBM MQ specifických pro vývoj aplikací:

Související pojmy

“Použití modulu MQI v klientské aplikaci” na stránce 875
Tato kolekce témat se zabývá rozdíly mezi zápisem vaší aplikace IBM MQ pro spuštění v prostředí klienta rozhraní MQI (MQI) a ke spuštění v prostředí úplného správce front produktu IBM MQ .

“Kanály-uživatelské programy pro kanály systému zpráv” na stránce 926
Tato kolekce témat obsahuje informace o kanálových programech typu IBM MQ pro kanály systému zpráv.

“Aspekty návrhu pro aplikace produktu IBM MQ” na stránce 43
Když se rozhodnete, jak vaše aplikace mohou využívat výhody platforem a prostředí, které máte k dispozici, musíte se rozhodnout, jak používat funkce nabízené produktem IBM MQ.

“Psaní procedurální aplikace pro řazení do fronty” na stránce 689
Prostřednictvím těchto informací můžete získat informace o vytváření aplikací pro řazení do front, připojování a odpojování od správce front, publikování a odběru a otevírání a zavírání objektů.

“Zápis procedurálních aplikací klienta” na stránce 874
Co potřebujete vědět, chcete-li psát klientské aplikace na serveru IBM MQ pomocí procedurálního jazyka.

“Vyvíjení aplikací MQI s produktem IBM MQ” na stránce 685
Produkt IBM MQ poskytuje podporu pro jazyky C, Visual Basic, COBOL, Assembler, RPG, pTALa PL/I. Tyto procedurální jazyky používají rozhraní fronty zpráv (MQI) pro přístup ke službám front zpráv.

“Objektově orientované aplikace” na stránce 10
Produkt IBM MQ poskytuje podporu pro produkty .NET, ActiveX, C + +, Javaa JMS. Tyto jazyky a rámce používají objektový model produktu IBM MQ , který poskytuje třídy, které poskytují stejné funkce jako volání a struktury produktu IBM MQ . Některé z jazyků a rámců, které používají model objektů produktu IBM MQ , poskytují další funkce, které nejsou k dispozici při použití procedurálních jazyků s rozhraním MQI (Message Queue Interface).

“použitíIBM MQ classes for JMS” na stránce 69
IBM MQ classes for Java Message Service (IBM MQ classes for JMS) je poskytovatel JMS , který je dodáván s IBM MQ. Stejně jako implementace rozhraní definovaných v balíku javax.jms poskytuje produkt IBM MQ classes for JMS dvě sady rozšíření rozhraní API produktu JMS .

“Použití rozhraní Model objektu Component Model (IBM MQ Automation Classes for ActiveX)” na stránce 561
Produkt IBM MQ Automation Classes for ActiveX (MQAX) jsou komponenty ActiveX , které poskytují třídy, které můžete použít ve své aplikaci pro přístup k produktu IBM MQ.

“použitíIBM MQ classes for Java” na stránce 308
Použijte IBM MQ v prostředí Java . IBM MQ classes for Java umožňuje aplikaci Java připojit se k IBM MQ jako klient IBM MQ nebo se připojit přímo ke správci front IBM MQ .

“Vývoj aplikací produktu .NET” na stránce 514
IBM MQ classes for .NET umožňuje programu zapsaným v programovacím rámci .NET pro připojení k serveru IBM MQ jako IBM MQ MQI client nebo k přímému připojení k serveru IBM MQ .

“Vývoj aplikací C++” na stránce 485
Produkt IBM MQ poskytuje třídy C + + ekvivalentní s objekty IBM MQ a některé další třídy ekvivalentní k datovým typům pole. Poskytuje řadu funkcí, které nejsou prostřednictvím rozhraní MQI k dispozici.

“Sestavení procedurální aplikace” na stránce 966
Aplikaci IBM MQ můžete psát v jednom z několika procedurálních jazyků a aplikaci spustit na několika různých platformách.

Související úlohy

“Vývoj webových služeb pomocí produktu IBM MQ” na stránce 1255
Můžete vyvíjet aplikace produktu IBM MQ pro webové služby pomocí přenosu IBM MQ pro protokol SOAP.

“Použití ukázkových procedurálních programů produktu IBM MQ” na stránce 1034
Tyto ukázkové programy jsou napsány v procedurálních jazycích a demonstrují typická použití rozhraní MQI (Message Queue Interface). Programy produktu IBM MQ na různých platformách.

Související informace

Scénáře transakčního podpory

Akce, které mohou vaše aplikace provádět

Můžete vyvinout aplikace k odesílání a přijímání zpráv, které potřebujete pro podporu svých obchodních procesů. Také můžete vyvíjet aplikace pro správu správců front a souvisejících prostředků.

Akce, které mohou vaše aplikace provádět s produktem IBM MQ for Multiplatforms

Multi

V systému Multiplatformymůžete psát aplikace, které provádějí následující akce:

- Odešlete zprávy do jiných aplikací spuštěných pod stejným operačním systémem. Aplikace mohou být buď ve stejném nebo v jiném systému.
- Odesílat zprávy aplikacím, které jsou spuštěny na jiných platformách produktu IBM MQ .
- Použijte řazení zpráv do fronty z prostředí CICS pro IBM i IBM i, TXSeries pro systémy AIX, HP-UX, Solarisa Windows .
- Použití řazení zpráv do fronty z prostředí Encina pro systémy AIX, HP-UX, Solarisa Windows .
- Zařazení do fronty zpráv z produktu Tuxedo pro systémy AIX, AT & T, HP-UX, Solarisa Windows .
- Použijte produkt IBM MQ jako správce transakcí koordinující aktualizace provedené externími správci prostředků v rámci jednotek práce produktu IBM MQ . Následující externí správci prostředků jsou podporováni a jsou v souladu s rozhraním XA sdružení X/OPEN
  - Db2
  - Informix
  - Oracle
  - Sybase
- Zpracovat několik zpráv dohromady jako jedinou jednotku práce, kterou lze potvrdit nebo zazálohovat.
- Spusťte z plného prostředí IBM MQ nebo spusťte prostředí klienta IBM MQ .

Akce, které mohou vaše aplikace provádět s produktem IBM MQ for z/OS

z/OS

V systému z/OSmůžete psát aplikace, které provádějí následující akce:

- Použijte řazení zpráv do fronty v rámci produktu CICS nebo IMS.
- Odesílat zprávy mezi dávkovými aplikacemi, produktem CICSa aplikacemi produktu IMS , výběrem nejvhodnějšího prostředí pro každou funkci.
- Odesílat zprávy aplikacím, které jsou spuštěny na jiných platformách produktu IBM MQ .
- Zpracovat několik zpráv dohromady jako jedinou jednotku práce, kterou lze potvrdit nebo zazálohovat.
- Odesílat zprávy do aplikací produktu IMS a pracovat s nimi prostřednictvím mostu IMS .
- Účastněte se jednotek práce koordinovaných službami RRS.

Každé prostředí v produktu z/OS má své vlastní charakteristiky, výhody a nevýhody. Výhoda produktu IBM MQ for z/OS spočívá v tom, že aplikace nejsou vázány na žádné prostředí, ale mohou být distribuovány tak, aby mohly využívat výhod každého prostředí. Například můžete vyvinout rozhraní koncového uživatele pomocí TSO nebo CICS, můžete spustit zpracování náročných modulů v dávkách z/OS a můžete spustit databázové aplikace v produktu IMS nebo CICS. Ve všech případech mohou různé části aplikace komunikovat pomocí zpráv a front.

Návrháři aplikací produktu IBM MQ si musí být vědomi rozdílů a omezení daných těmito prostředími. Příklad:

- Produkt IBM MQ poskytuje funkce, které umožňují vzájemnou komunikaci mezi správci front (nazývám to *distribuované řazení do fronty*).
- Metody potvrzování a provádění změn se liší mezi dávkovými a CICS prostředími.
- Produkt IBM MQ for z/OS poskytuje podporu v prostředí IMS pro online programy zpracování zpráv (MPP), interaktivní programy rychlých cest (IFPs) a programy pro zpracování dávkových zpráv (BMP). Pokud zapisujete dávkové programy DL/I, postupujte podle pokynů uvedených v tématech jako “Sestavení dávkových aplikací produktu z/OS” na stránce 1001 a “Aspekty dávky produktu z/OS” na stránce 701 pro dávkové programy z/OS .
- Ačkoli může v jednom systému z/OS existovat více instancí produktu IBM MQ for z/OS , může se region CICS připojit k jednomu správci front najednou. Do stejného správce front lze však připojit více než jeden region produktu CICS . V dávkových prostředích IMS a z/OS se mohou programy připojovat k více než jednomu správci front.
- Produkt IBM MQ for z/OS umožňuje sdílení lokálních front skupinou správců front s vyšší propustností a dostupností. Tyto fronty se nazývají *sdílené fronty*a správci front tvoří *skupinu sdílení front*, která může zpracovávat zprávy ve stejných sdílených frontách. Dávkové aplikace se mohou připojit k jednomu z několika správců front v rámci skupiny sdílení front uvedením názvu skupiny sdílení front, nikoli konkrétního názvu správce front. To se označuje jako *připojeno k dávkovému zpracování skupiny*, nebo více jednoduše *skupinového připojení*. Viz Sdílené fronty a skupiny sdílení front.

z/OS Rozdíly mezi podporovanými prostředími a jejich omezeními jsou vysvětleny dále v tématu “Použití a zápis aplikací v systému IBM MQ for z/OS” na stránce 851.

Související pojmy

“Koncepty vývoje aplikací” na stránce 6
K zápisu aplikací IBM MQ můžete použít volbu procedurálních nebo objektově orientovaných jazyků. Odkazy v tomto tématu použijte pro informace o koncepcích produktu IBM MQ , které jsou užitečné pro vývojáře aplikací.

“Aspekty návrhu pro aplikace produktu IBM MQ” na stránce 43
Když se rozhodnete, jak vaše aplikace mohou využívat výhody platforem a prostředí, které máte k dispozici, musíte se rozhodnout, jak používat funkce nabízené produktem IBM MQ.

“Psaní procedurální aplikace pro řazení do fronty” na stránce 689
Prostřednictvím těchto informací můžete získat informace o vytváření aplikací pro řazení do front, připojování a odpojování od správce front, publikování a odběru a otevírání a zavírání objektů.

“Zápis procedurálních aplikací klienta” na stránce 874
Co potřebujete vědět, chcete-li psát klientské aplikace na serveru IBM MQ pomocí procedurálního jazyka.

“použitíIBM MQ classes for JMS” na stránce 69
IBM MQ classes for Java Message Service (IBM MQ classes for JMS) je poskytovatel JMS , který je dodáván s IBM MQ. Stejně jako implementace rozhraní definovaných v balíku javax.jms poskytuje produkt IBM MQ classes for JMS dvě sady rozšíření rozhraní API produktu JMS .

“Použití rozhraní Model objektu Component Model (IBM MQ Automation Classes for ActiveX)” na stránce 561
Produkt IBM MQ Automation Classes for ActiveX (MQAX) jsou komponenty ActiveX , které poskytují třídy, které můžete použít ve své aplikaci pro přístup k produktu IBM MQ.

“použitíIBM MQ classes for Java” na stránce 308
Použijte IBM MQ v prostředí Java . IBM MQ classes for Java umožňuje aplikaci Java připojit se k IBM MQ jako klient IBM MQ nebo se připojit přímo ke správci front IBM MQ .

“Vývoj aplikací produktu .NET” na stránce 514
IBM MQ classes for .NET umožňuje programu zapsaným v programovacím rámci .NET pro připojení k serveru IBM MQ jako IBM MQ MQI client nebo k přímému připojení k serveru IBM MQ .

“Vývoj aplikací Microsoft Windows Communication Foundation (WCF) s IBM MQ” na stránce 1217
Vlastní kanál Microsoft Windows Communication Foundation (WCF) pro produkt IBM MQ odesílá a přijímá zprávy mezi klienty WCF a službami.

“Vývoj aplikací C++” na stránce 485

Produkt IBM MQ poskytuje třídy C + + ekvivalentní s objekty IBM MQ a některé další třídy ekvivalentní k datovým typům pole. Poskytuje řadu funkcí, které nejsou prostřednictvím rozhraní MQI k dispozici.

“Sestavení procedurální aplikace” na stránce 966
Aplikaci IBM MQ můžete psát v jednom z několika procedurálních jazyků a aplikaci spustit na několika různých platformách.

Související úlohy

“Použití ukázkových procedurálních programů produktu IBM MQ” na stránce 1034
Tyto ukázkové programy jsou napsány v procedurálních jazycích a demonstrují typická použití rozhraní MQI (Message Queue Interface). Programy produktu IBM MQ na různých platformách.

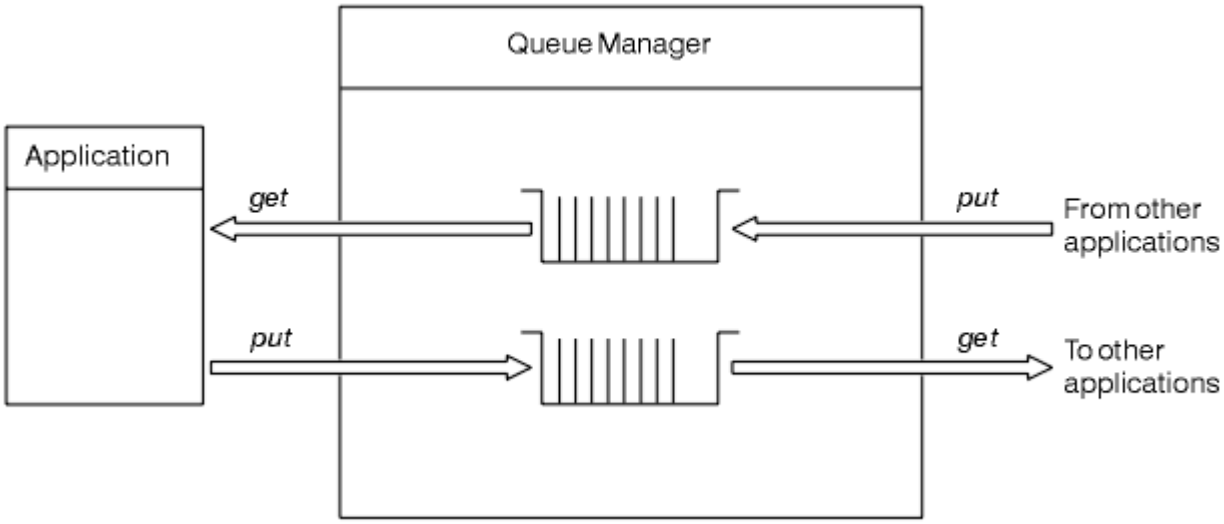
Související informace

Zabezpečení

Aplikační programy používající rozhraní MQI

Aplikační programy produktu IBM MQ potřebují určité objekty, než je možné úspěšně spustit.

Produkt Obrázek 1 na stránce 10 zobrazuje aplikaci, která odebírá zprávy z fronty, zpracuje je a poté odešle některé výsledky do jiné fronty ve stejném správci front.



Obrázek 1. Fronty, zprávy a aplikace

Zatímco aplikace mohou vkládat zprávy do lokálních nebo vzdálených front (pomocí příkazu MQPUT), mohou zprávy získat pouze přímo z lokálních front (pomocí příkazu MQGET).

Před spuštěním této aplikace musí být splněny následující podmínky:

- Správce front musí existovat a musí být spuštěn.
- Musí být definována první aplikační fronta, ze které se zprávy mají odstranit.
- Musí být definována také druhá fronta, na které aplikace vkládá zprávy.
- Aplikace musí být schopna připojit se ke správci front. Chcete-li tak učinit, musí být propojena s IBM MQ. Viz “Sestavení procedurální aplikace” na stránce 966.
- Aplikace, které vložila zprávy do první fronty, se musí také připojit ke správci front. Jsou-li vzdálené, musí být také nastaveny pomocí přenosových front a kanálů. Tato část systému není zobrazena v Obrázek 1 na stránce 10.

Objektově orientované aplikace

Produkt IBM MQ poskytuje podporu pro produkty .NET, ActiveX, C + +, Javaa JMS. Tyto jazyky a rámce používají objektový model produktu IBM MQ , který poskytuje třídy, které poskytují stejné funkce jako volání a struktury produktu IBM MQ . Některé z jazyků a rámců, které používají model objektů produktu

IBM MQ , poskytují další funkce, které nejsou k dispozici při použití procedurálních jazyků s rozhraním MQI (Message Queue Interface).

Podrobnosti o třídách, metodách a vlastnostech poskytovaných tímto modelem lze najít v dokumentu “Objektový model produktu IBM MQ” na stránce 11.

.NET
Informace o programování programů .NET pomocí tříd IBM MQ .NET viz Vývoj aplikací .NET . Klienti Message Service Clients for C/C++ a .NET poskytují rozhraní API (Application Programming Interface) s názvem XMS , které má stejnou sadu rozhraní jako Java Message Service (JMS) API.

ActiveX
IBM MQ ActiveX je obecně znám jako MQAX. Aplikace MQAX je zahrnuta jako součást produktu IBM MQ for Windows. Podpora pro ActiveX byla stabilizována na úrovni IBM WebSphere MQ 6.0 . Informace o kódujících programech pomocí modelu objektu IBM MQ v ActiveX Použití rozhraní modelu objektu komponenty (WebSphere MQ Automation Classes for ActiveX).

V 9.0.0 V produktu IBM MQ 9.0je podpora produktu MicrosoftActive X zamítnuta. Třídy IBM MQ pro .NET jsou doporučenými náhradními technologiemi. Další informace viz Vývoj aplikací .NET.

JAZYK C++
Produkt IBM MQ poskytuje třídy C + + ekvivalentní s objekty IBM MQ a některé další třídy ekvivalentní k datovým typům pole. Poskytuje řadu funkcí, které nejsou prostřednictvím rozhraní MQI k dispozici. See Použití C++ for information about coding programs using the IBM MQ Object Model in C++. Message Service Clients for C/C++ and .NET provide an application programming Interface (API) called XMS that has the same set of interfaces as the Java Message Service (JMS) API.

Java
Informace o kódujících programech pomocí modelu objektu IBM MQ v produktu Javaviz Použití modulu IBM MQ classes for Java . Produkt IBM nebude provádět žádná další vylepšení produktu IBM MQ classes for Java a jsou funkčně stabilizováni na úrovni dodávané v produktu IBM MQ 8.0. Informace o rozdílech mezi IBM MQ classes for Java a IBM MQ classes for JMS vám pomohou se rozhodnout, která z nich použít, viz “Volba použití produktu IBM MQ classes for Java nebo IBM MQ classes for JMS” na stránce 46.

JMS
IBM MQ také poskytuje třídy, které implementují specifikaci Java Message Service (JMS). Podrobnosti o produktu IBM MQ classes for JMSnaleznete v tématu Použití modulu IBM MQ classes for JMS. Informace o rozdílech mezi IBM MQ classes for Java a IBM MQ classes for JMS vám pomohou rozhodnout se, která z nich se mají použít, viz “Volba použití produktu IBM MQ classes for Java nebo IBM MQ classes for JMS” na stránce 46.

IBM Message Service Client for C/C++ a IBM Message Service Client for .NET poskytují rozhraní API s názvem XMS , které má stejnou sadu rozhraní jako Java Message Service (JMS) API. Další informace viz Úvod do produktu IBM Message Service Client for .NET.

Související pojmy

“Vyvíjení aplikací MQI s produktem IBM MQ” na stránce 685
Produkt IBM MQ poskytuje podporu pro jazyky C, Visual Basic, COBOL, Assembler, RPG, pTALa PL/I. Tyto procedurální jazyky používají rozhraní fronty zpráv (MQI) pro přístup ke službám front zpráv.

“Koncepty vývoje aplikací” na stránce 6
K zápisu aplikací IBM MQ můžete použít volbu procedurálních nebo objektově orientovaných jazyků. Odkazy v tomto tématu použijte pro informace o koncepcích produktu IBM MQ , které jsou užitečné pro vývojáře aplikací.

Související informace

Technický přehled
Odkaz na vývoj aplikací

Objektový model produktu IBM MQ

Model objektu IBM MQ se skládá ze tříd, metod a vlastností.

Model objektu produktu IBM MQ se skládá z následujících položek:

- *Třídy* představující známé koncepce produktu IBM MQ , jako jsou správci front, fronty a zprávy.
- *Metody* pro každou třídu odpovídající voláním MQI.
- *Vlastnosti* na každé třídě odpovídající atributům objektů IBM MQ .

Při vytváření aplikace IBM MQ s použitím modelu objektu produktu IBM MQ vytváříte instance těchto tříd v aplikaci. Instance třídy v objektově orientovaném programování se nazývá *objekt*. Když byl objekt vytvořen, můžete s ním interagovat tím, že zkontrolujete nebo nastavíte hodnoty vlastností objektu (ekvivalent volání MQINQ nebo MQSET) a voláním metody proti objektu (což je ekvivalent k zadání dalších volání MQI).

Třídy

Model objektu IBM MQ poskytuje následující základní sadu tříd.

Skutečná implementace modelu se mírně liší mezi různými podporovanými objektově orientovaným prostředím.

MQQueueManager
Objekt třídy MQQueueManager představuje připojení ke správci front. Má metody Connect (), Disconnect (), Commit () a Backout () (ekvivalentní MQCONN nebo MQCONNX, MQDISC, MQCMIT a MQBACK). Má vlastnosti odpovídající atributům správce front. Přístup k vlastnosti atributu správce front se implicitně připojuje ke správci front, pokud již není připojen. Destrojení objektu MQQueueManager se implicitně odpojí od správce front.

MQQUEUE
Objekt třídy MQQueue představuje frontu. Má metody Put () a Get () do fronty a z fronty (ekvivalent MQPUT a MQGET). Má vlastnosti odpovídající atributům fronty. Při přístupu k vlastnosti atributu fronty nebo při volání metody Put () nebo Get () se implicitně otevře fronta (ekvivalent MQOPEN). Destrojení objektu MQQueue implicitně zavírá frontu (ekvivalent MQCLOSE).

MQTopic
Objekt třídy MQTopic představuje téma. Má metody vložení () (publikování) a získání () (příjem nebo odběr) zpráv do a z daného tématu (ekvivalent MQPUT a MQGET). Má vlastnosti odpovídající atributům tématu. K objektu MQTopic lze přistupovat pouze pro publikování nebo odběr, nikoli pro obě současně. Je-li použit pro příjem zpráv, lze objekt MQTopic vytvořit s nespravovaným nebo spravovaným odběrem a jako trvalý nebo netrvalý odběratel-je pro tyto různé scénáře poskytnuto více přetížených konstruktorů.

Zpráva MQMessage
Objekt třídy MQMessage představuje zprávu, která má být vložena do fronty nebo z fronty. Obsahuje vyrovnávací paměť a zapouzdřuje jak aplikační data, tak MQMD. Má vlastnosti odpovídající polím MQMD a metodám, které umožňují zapisovat a číst uživatelská data různých typů (například řetězce, dlouhá celá čísla, krátká celá čísla, jednotlivé bajty) do vyrovnávací paměti a z ní.

Volby MQPutMessage
Objekt třídy voleb MQPutMessagepředstavuje strukturu MQPMO. Má vlastnosti odpovídající polím MQPMO.

Volby MQGetMessage
Objekt třídy Volby MQGetMessagepředstavuje strukturu MQGMO. Má vlastnosti odpovídající polím MQGMO.

Proces MQProcess
Objekt třídy MQProcess představuje definici procesu (používá se se spouštěním). Má vlastnosti, které představují atributy definice procesu.

MQDistributionList
Objekt třídy MQDistributionList představuje distribuční seznam (používá se k odeslání více zpráv s jedním MQPUT). Obsahuje seznam objektů položek MQDistributionList.

Multi **Položka MQDistributionList**

Objekt třídy položek MQDistributionListpředstavuje jediné místo určení distribučního seznamu. Zapouzdřuje struktury MQOR, MQRR a MQPMR a má vlastnosti odpovídající polím těchto struktur.

odkazy na objekty

V programu IBM MQ , který používá rozhraní MQI, vrací produkt IBM MQ pro tento program obslužné rutiny připojení a obslužné rutiny objektů.

Tyto popisovače musí být předány jako parametry při následných voláních IBM MQ . S modelem objektu produktu IBM MQ jsou tyto popisovače skryty před aplikačním programem. Místo toho se vytvoření objektu z třídy vede k odkazu na objekt, který se vrací do aplikačního programu. Jedná se o odkaz na objekt, který se používá při vytváření volání metody a přístupu k vlastnostem pro objekt.

Návratové kódy

Zadání metody volání metody nebo nastavení hodnoty vlastnosti má za následek nastavení návratových kódů.

Tyto návratové kódy jsou kódem dokončení a kódem příčiny a jsou samy vlastnostmi objektu. Hodnoty kódu dokončení a kódu příčiny jsou stejné jako hodnoty definované pro rozhraní MQI, s některými dodatečnými hodnotami specifickými pro objektově orientované prostředí.

Zprávy produktu IBM MQ

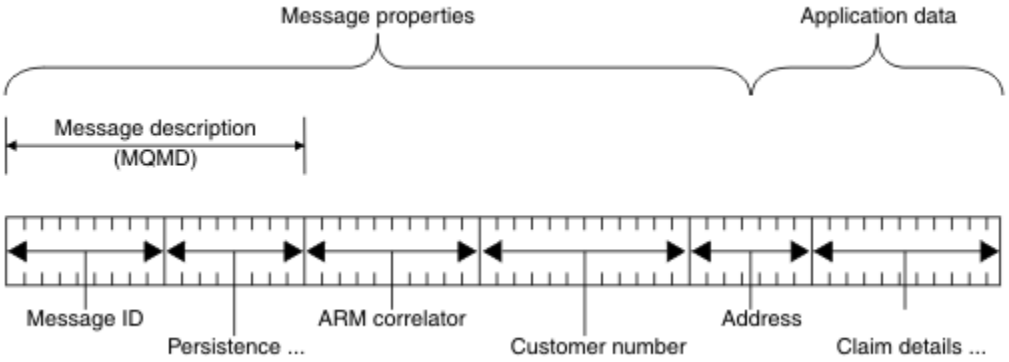
Zpráva IBM MQ se skládá z vlastností zpráv a aplikačních dat. Deskriptor zpráv pro řazení zpráv do front (MQMD) obsahuje řídicí informace, které jsou připojeny k datům aplikace při přenosu zprávy mezi odesílající a přijímající aplikací.

Části zprávy

Zprávy IBM MQ se skládají ze dvou částí:

- Vlastnosti zprávy
- Data aplikace

Obrázek 2 na stránce 13 představuje zprávu a ukazuje, jak je logicky rozdělena na vlastnosti zprávy a aplikační data.



Obrázek 2. Zastupování zprávy

Data aplikace, která jsou přenášeno ve zprávě IBM MQ , se nezmění správcem front, pokud se k němu neprovedou převod dat. Produkt IBM MQ také nevloží žádná omezení týkající se obsahu těchto dat. Délka dat v každé zprávě nesmí překročit hodnotu atributu **MaxMsgLength** jak ve frontě, tak i ve správci front.

ULW V systému UNIX, Linux®, and Windowsje atribut *MaxMsgLength* správce front a fronta standardně 4 MB (4 194 304 bajtů), který lze v případě potřeby změnit až na maximální hodnotu 100 MB (104 857 600 bajtů).

IBM i V systému IBM ije atribut *MaxMsgLength* správce front a fronta standardně 4 MB (4 194 304 bajtů), který lze v případě potřeby změnit až na maximální hodnotu 100 MB (104 857 600 bajtů). Máte-li v úmyslu používat zprávy produktu IBM MQ větší než 15 MB v systému IBM i, viz “Sestavuje se vaše procedurální aplikace na IBM i” na stránce 984.

z/OS V systému z/OSje atribut **MaxMsgLength** správce front nastaven na hodnotu 100 MB a atribut **MaxMsgLength** fronty je standardně nastaven na 4 MB (4 194 304 bajtů), který lze podle potřeby změnit až na maximální hodnotu 100 MB.

V některých případech můžete zprávy mírně zkrátit, než je hodnota atributu **MaxMsgLength** . Další informace viz “Data ve zprávě” na stránce 727.

Zprávu vytvoříte, když použijete volání MQPUT nebo MQPUT1 MQI. Jako vstup těchto volání zadáte řídicí informace (jako je priorita zprávy a jméno fronty odpovědí) a vaše data, a volání pak vloží zprávu do fronty. Další informace o těchto voláních viz MQPUT a MQPUT1 .

deskriptor zprávy

K informacím o řízení zpráv můžete přistupovat pomocí struktury MQMD, která definuje *deskriptor zprávy*.

Úplný popis struktury MQMD naleznete v tématu Deskriptor MQMD-Message.

Popis způsobu použití polí v rámci MQMD, které obsahují informace o původu zprávy, naleznete v tématu “kontext zprávy” na stránce 41.

Jsou zde různé verze deskriptoru zpráv. Další informace o seskupování a segmentování zpráv (viz “Skupiny zpráv” na stránce 38) jsou poskytovány ve verzi 2 deskriptoru zpráv (nebo MQMDE). To je stejné jako deskriptor zprávy verze 1, ale má další pole. Tato pole jsou popsána v MQMMDE-rozšíření deskriptoru zpráv.

Typy zpráv

Existují čtyři typy zpráv definované produktem IBM MQ.

Tyto čtyři zprávy jsou:

- Datagram
- Zprávy požadavků
- Zprávy odpovědí
- Zprávy sestav
  - Typy zpráv sestav
  - Volby zprávy sestavy

Aplikace mohou používat první tři typy zpráv k předávání informací mezi sebou. Čtvrtý typ, sestava, je určena pro aplikace a správce front, které mají být použity při hlášení informací o událostech, jako je například výskyt chyby.

Každý typ zprávy je identifikován hodnotou MQMT_ *. Můžete také definovat své vlastní typy zpráv. Rozsah hodnot, které můžete použít, viz MsgType.

Datagramy

Použijte *datagram* , když nevyžadujete odpověď od aplikace, která přijme zprávu (to znamená, že obdrží zprávu z fronty).

Příkladem aplikace, která může používat datagramy, je ta, která zobrazuje informace o letu v salónku na letišti. Zpráva může obsahovat data pro celou obrazovku letových informací. Je nepravděpodobné, že by

taková aplikace požádala o potvrzení o přijetí zprávy, protože pravděpodobně nezáleží na tom, zda zpráva nebyla doručena. Aplikace po krátké době odešle aktualizační zprávu.

Zprávy požadavků

Použijte *zprávu požadavku* , chcete-li odpověď z aplikace, která přijímá zprávu.

Příkladem aplikace, která může používat zprávy požadavků, je taková aplikace, která zobrazuje zůstatek kontrolního účtu. Zpráva požadavku by mohla obsahovat číslo účtu a zpráva odpovědi by obsahovala zůstatek na účtu.

Chcete-li propojit svou zprávu s odpovědí se zprávou požadavku, existují dvě možnosti:

- Učiňte aplikaci, která zpracovává zprávu požadavku zodpovědnou za zajištění, že vloží informace do zprávy odpovědi, která se vztahuje ke zprávě požadavku.
- Pole sestavy v deskriptoru zprávy ve zprávě požadavku slouží k určení obsahu polí *MsgId* a *CorrelId* ve zprávě odpovědi:
  - Můžete požádat o zkopírování buď *MsgId* , nebo `CorrelId` z původní zprávy do pole *CorrelId* zprávy odpovědi (výchozí akce je zkopírování *MsgId*).
  - Můžete požadovat, aby se pro zprávu odpovědi generovala buď nová `MsgId` , nebo že *MsgId* původní zprávy se má zkopírovat do pole *MsgId* zprávy odpovědi (výchozí akce je generovat nový identifikátor zprávy).

Odpovědi na zprávy

Když odpovíte na jinou zprávu, použijte *zprávu odpovědi* .

Při vytváření zprávy s odpovědí respektují všechny volby, které byly nastaveny v deskriptoru zprávy pro zprávu, na kterou odpovídáte. Volby sestavy určují obsah polí identifikátoru zprávy (`MsgId`) a identifikátoru korelace (`CorrelId`). Tato pole umožňují aplikaci, která přijímá odpověď, aby korelovala odpověď se svým původním požadavkem.

Hlášení zpráv

Zprávy sestav informují aplikace o událostech, jako je například výskyt chyby při zpracování zprávy.

Mohou být generovány pomocí:

- správce front,
- agent kanálu zpráv (například, pokud nemůže doručit zprávu), nebo
- Aplikace (například, pokud ji nemůže použít data ve zprávě).

Zprávy sestavy mohou být generovány kdykoli a mohou přicházet do fronty, když je vaše aplikace neočekává.

Typy zpráv sestav

Když vložíte zprávu do fronty, můžete si vybrat, zda chcete přijmout:

- *Zpráva o výjimce*. Toto je odesláno jako odpověď na zprávu s nastaveným příznakem výjimek. Je generován agentem kanálu zpráv (MCA) nebo aplikací.
- *Zpráva o vypršení platnosti zprávy*. To označuje, že se aplikace pokusila načíst zprávu, která dosáhla své prahové hodnoty vypršení platnosti; zpráva je označena jako vyřazená. Tento typ sestavy je generován správcem front.
- *Potvrzení zprávy o příjmu (COA)*. Tato zpráva informuje o tom, že zpráva dosáhla cílové fronty. Je generován správcem front.
- *Potvrzení o potvrzení doručení (COD)*. To znamená, že zpráva byla načtena přijímající aplikací. Je generován správcem front.

- *Zpráva sestavy s kladným oznámením akce (PAN)*. To znamená, že požadavek byl úspěšně obsloužen (to znamená, že akce požadovaná ve zprávě byla úspěšně provedena). Tento typ sestavy je generován aplikací.
- *Zpráva sestavy Negative action notification (NAN)*. To znamená, že požadavek nebyl úspěšně obsloužen (to znamená, že akce požadovaná ve zprávě nebyla úspěšně provedena). Tento typ sestavy je generován aplikací.

Poznámka: Každý typ zprávy sestavy obsahuje jednu z následujících možností:

- Celá původní zpráva
- Prvních 100 bajtů dat v původní zprávě
- Žádná data z původní zprávy

Když vložíte zprávu do fronty, můžete požadovat více než jeden typ zprávy sestavy. Vyberete-li zprávu s potvrzením doručení zprávy a volby zprávy sestavy výjimek, obdržíte zprávu o výjimce, pokud zpráva selže. Pokud však vyberete pouze volbu zprávy s potvrzením doručení zprávy a zpráva se nepodaří doručit, nedostanete zprávu hlášení o výjimce.

Zprávy, které požadujete, jsou-li splněna kritéria pro generování konkrétní zprávy, jsou jediní, které jste obdrželi.

Volby zprávy sestavy

Po vzniku výjimky můžete *zahodit* zprávu. Pokud vyberete volbu vyřazení a vyžádali jste zprávu s hlášením o výjimce, zpráva se odešle do *ReplyToQ* a *ReplyToQMgr*a původní zpráva se vyřadí.

Poznámka: Výhodou této funkce je, že můžete snížit počet zpráv, které se budou do fronty nedoručených zpráv odpracovat. Znamená to však, že vaše aplikace, pokud neodešle pouze datagramové zprávy, se musí vypořádat s vrácenými zprávami. Je-li generována zpráva o výjimce, zdědí perzistenci původní zprávy.

Pokud zprávu sestavy nelze doručit (je-li fronta například plná), bude zpráva sestavy umístěna do fronty nedoručených zpráv.

Chcete-li přijmout zprávu sestavy, zadejte název fronty pro odpověď do pole *ReplyToQ* ; v opačném případě dojde k selhání MQPUT nebo MQPUT1 původní zprávy s MQRC_MISSING_REPLY_TO_Q.

Můžete použít jiné volby sestavy v deskriptoru zpráv (MQMD) zprávy k určení obsahu polí *MsgId* a *CorrelId* všech zpráv sestavy, které jsou vytvořeny pro zprávu:

- Můžete požadovat zkopírování buď *MsgId* nebo *CorrelId* z původní zprávy do pole *CorrelId* zprávy sestavy. Výchozí akce je kopírovat identifikátor zprávy. Použijte MQRO_COPY_MSG_ID_TO_CORRELID, protože umožňuje odesílateli zprávy korelovat zprávu odpovědi nebo zprávy s původní zprávou. Identifikátor korelace odpovědi nebo zprávy sestavy je identický s identifikátorem zprávy původní zprávy.
- Můžete požadovat, aby se pro zprávu sestavy generovala buď nová *MsgId* , nebo že se *MsgId* z původní zprávy má zkopírovat do pole *MsgId* zprávy sestavy. Předvolená akce je generovat nový identifikátor zprávy. Použijte MQRO_NEW_MSG_ID, protože zajišťuje, že každá zpráva v systému má jiný identifikátor zprávy a může být jednoznačně odlišena od všech ostatních zpráv v systému.
- Speciální aplikace mohou vyžadovat použití hodnoty MQRO_PASS_MSG_ID nebo MQRO_PASS_CORREL_ID. Musíte však navrhnout aplikaci, která čte zprávy z fronty, aby zajistila, že funguje správně, když například fronta obsahuje více zpráv se stejným identifikátorem zprávy.

  Serverové aplikace musí zkontrolovat nastavení těchto příznaků ve zprávě požadavku a v odpovídajícím způsobem nastavit pole *MsgId* a *CorrelId* ve zprávě odpovědi nebo sestavy.

  Aplikace, které se chovají jako prostředníci mezi aplikací žadatele a serverovou aplikací, nekontrolují nastavení těchto parametrů. Důvodem je to, že tyto aplikace obvykle potřebují předat zprávu do serverové aplikace s poli *MsgId*, *CorrelId*a *Report* nezměněná. To umožní serverovou aplikaci kopírovat *MsgId* z původní zprávy do pole *CorrelId* zprávy odpovědi.

Při generování sestavy o zprávě se musí aplikace serveru otestovat, aby bylo možné zjistit, zda byla některá z těchto voleb nastavena.

Další informace o tom, jak používat zprávy sestav, najdete v tématu Sestava.

Ke značkování povahy sestavy používají správci front rozsah kódů zpětné vazby. Tyto kódy umístí do pole *Feedback* v deskriptoru zpráv ve zprávě sestavy. Správci front mohou také v poli *Feedback* vrátit kódy příčiny MQI. IBM MQ definuje rozsah kódů zpětné vazby pro aplikace, které mají být použity.

Další informace o zpětné vazbě a kódech příčiny najdete v tématu Zpětná vazba.

Příkladem programu, který by mohl použít kód zpětné vazby, je takový, který monitoruje pracovní zátěže jiných programů obsluhujících frontu. Pokud existuje více než jedna instance programu obsluhujícího frontu a počet zpráv přicházejících do fronty již neopravňuje tento program, takový program může odeslat zprávu s hlášením (s kódem zpětné vazby MQFB_QUIT) jednomu z obsluhujících programů, aby indikoval, že by program měl ukončit svou činnost. (Monitorovací program by mohl použít volání MQINQ k vyhledání toho, kolik programů obsluhují frontu.)

Multi *Sestavy a segmentované zprávy*

Nepodporováno na IBM MQ for z/OS.

Je-li zpráva segmentována a žádáte o generování sestav, můžete obdržet více sestav, než byste měli, kdyby nebyla zpráva segmentována.

Popis segmentovaných zpráv viz “Segmentace zpráv” na stránce 759.

Pro sestavy generované produktem IBM MQ

Pokud jste své zprávy segmentoval nebo umožnili správci front tak učinit, existuje pouze jeden případ, kdy můžete očekávat, že obdržíte jednu sestavu pro celou zprávu. To znamená, že jste požadovali pouze sestavy COD a v aplikaci získání jste zadali MQGMO_COMPLETE_MSG.

V jiných případech musí být vaše aplikace připravena se vypořádat s několika sestavami; obvykle jedna pro každý segment.

Poznámka: Pokud segmentujete zprávy a budete potřebovat pouze prvních 100 bajtů původních dat zprávy, změňte nastavení voleb sestavy tak, aby se žádal o sestavy bez dat pro segmenty s offsetem 100 nebo více. Pokud toto neuděláte a necháte nastavení tak, aby každý segment požaduje 100 bajtů dat, a načtete zprávy sestavy s jedinou MQGET specifikující MQGMO_COMPLETE_MSG, sestavy se sestaví do velké zprávy obsahující 100 bajtů načtených dat na každém odpovídajícím posunutí. Pokud k tomu dojde, potřebujete velkou vyrovnávací paměť nebo je třeba zadat MQGMO_ACCEPT_TRUNCATED_MSG.

Pro sestavy generované aplikacemi

Pokud vaše aplikace generuje sestavy, vždy zkopírujte záhlaví IBM MQ , která jsou přítomná na začátku původních dat zprávy, do dat zprávy sestavy.

Pak nepřidávejte žádný, 100 bajtů nebo všechny původní data zprávy (nebo jakoukoli jinou částku, kterou byste obvykle zahrnuli) do dat zprávy hlášení.

Záhlaví IBM MQ , která musí být kopírována, poznáte podle názvů po sobě jdoucími názvy formátu, počínaje MQMD a pokračuje se všemi přítomnými záhlavími. Následující názvy `Format` označují tato záhlaví IBM MQ :

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

MQH* označuje libovolný název, který začíná znaky MQH.

Název `Format` se vyskytuje na specifických pozicích pro MQDLH a MQXQH, ale pro ostatní záhlaví IBM MQ se vyskytuje na stejné pozici. Délka záhlaví je obsažena v poli, které se vyskytuje také na stejné pozici pro záhlaví MQMDE, MQIMSa všechny hlavičky MQH*.

Používáte-li produkt MQMD verze 1 a hlásíte-li se na segment nebo zprávu ve skupině nebo zprávu, jejíž segmentace je povolena, musí data sestavy začínat řetězcem MQMDE. Nastavte pole *OriginalLength* na délku původních dat zprávy s vyloučením délek všech záhlaví IBM MQ , které najdete.

Načítání sestav

Pokud požádáte o sestavy COA nebo COD, můžete požádat, aby byly pro vás znovu složeny s MQGMO_COMPLETE_MSG.

Příkaz MQGET s MQGMO_COMPLETE_MSG je uspokojen, když je ve frontě přítomno dostatek zpráv hlášení (jednoho typu, například COA, a se stejným *GroupId*), aby znázornil jednu úplnou původní zprávu. To platí i v případě, že samotné zprávy sestavy neobsahují úplná původní data; pole *OriginalLength* v každé zprávě sestavy udává délku původních dat reprezentovaných touto zprávou sestavy, i když samotná data nejsou k dispozici.

Tuto techniku můžete použít i v případě, že ve frontě existuje několik různých typů sestav (například COA a COD), protože příkaz MQGET s MQGMO_COMPLETE_MSG znovu sestaví zprávy sestav pouze v případě, že mají stejný kód *Feedback* . Tuto techniku však nelze obvykle použít pro hlášení výjimek, protože obecně mají odlišné kódy *Feedback* .

Tuto techniku můžete použít k získání kladné indikace, že byla doručena celá zpráva. Avšak ve většině případů je třeba vyhovět možnosti, že některé segmenty dorazí, zatímco jiné by mohly generovat výjimku (nebo vypršení platnosti, pokud jste to umožnili). V tomto případě nemůžete použít MQGMO_COMPLETE_MSG, protože obecně byste mohli získat různé kódy *Feedback* pro různé segmenty a pro segment byste mohli získat více než jednu sestavu. Můžete však použít funkce MQGMO_ALL_SEGMENTS_AVAILABLE.

Chcete-li tuto možnost povolit, budete možná muset načítat sestavy, jakmile dorazí, a sestavit obrázek ve vaší aplikaci toho, co se stalo s původní zprávou. Pole *GroupId* ve zprávě sestavy můžete použít ke korelaci sestav s *GroupId* původní zprávy a pole *Feedback* pro identifikaci typu každé zprávy sestavy. Způsob, jakým to provedete, závisí na požadavcích aplikace.

Jeden přístup je následující:

- Dotázat se na sestavy COD a zprávy o výjimkách.
- Po určité době zkontrolujte, zda byla přijata kompletní sada sestav COD pomocí MQGMO_COMPLEE_MSG. Je-li tomu tak, vaše aplikace ví, že byla zpracována celá zpráva.
- Pokud nejsou k dispozici žádné zprávy o výjimce týkající se této zprávy, problém se ošetří jako u nesegmentovaných zpráv, ale ujistěte se, že jste v určitém bodě vyčistili osiřelé segmenty.
- Pokud existují segmenty, pro které neexistují žádné sestavy libovolného druhu, mohou původní segmenty (nebo sestavy) čekat na opětovné připojení kanálu, nebo může být síť v určitém bodě přetížená. Pokud nebyly přijaty žádné zprávy o výjimkách (nebo pokud si myslíte, že ty, které máte, mohou být pouze dočasné), můžete se rozhodnout, že vaše aplikace bude čekat o něco déle.

  Stejně jako předtím je to podobné úvahám, které máte při práci s nesegmentovanými zprávami, kromě toho, že byste měli zvážit také možnost vyčištění osiřelých segmentů.

Není-li původní zpráva kritická (například, pokud se jedná o dotaz nebo zprávu, kterou lze později opakovat), nastavte dobu vypršení platnosti, abyste se ujistili, že osiřelé segmenty budou odebrány.

Správci front nižší úrovně

Když je sestava generována správcem front, který podporuje segmentaci, ale je přijat ve správci front, který nepodporuje segmentaci, struktura MQMDE (která identifikuje sestavu *Offset* a *OriginalLength* představovanou sestavou) je vždy zahrnuta do dat sestavy, kromě nuly, 100 bajtů nebo všech původních dat ve zprávě.

Pokud však segment zprávy prochází správcem front, který nepodporuje segmentaci, je struktura MQMDE v původní zprávě považována za data čistě jako data. Není tedy zahrnuta do dat sestavy, pokud bylo vyžádáno nula bajtů původních dat. Bez hodnoty MQMDE nemusí být zpráva sestavy užitečná.

Požadujte alespoň 100 bajtů dat v sestavách, pokud existuje možnost, že by zpráva mohla cestovat přes správce front nižší úrovně.

Formát dat řízení zpráv a dat zprávy

Správce front má zájem pouze o formát řídicích informací ve zprávě, zatímco aplikace obsluhující zprávy mají zájem o formát řídicích informací i dat.

Formát informací o řízení zpráv

Řídicí informace ve znakovém řetězcovém poli deskriptoru zpráv musí být ve znakové sadě používané správcem front.

Tuto znakovou sadu definuje atribut **CodedCharSetId** objektu správce front. Řídicí informace musí být v této znakové sadě, protože když aplikace předávají zprávy z jednoho správce front do jiného, agenti kanálu zpráv, kteří předávají zprávy, používají hodnotu tohoto atributu k určení toho, jaký převod dat má provést.

Formát dat zprávy

Můžete určit kteroukoli z následujících možností:

- Formát dat aplikace
- Znaková sada znakových dat
- Formát číselných dat

Chcete-li to provést, použijte tato pole:

Format
Toto indikuje přijímači zprávy formát dat aplikace ve zprávě.

Když správce front vytvoří zprávu, za určitých okolností použije pole *Format* k identifikaci formátu této zprávy. Pokud například správce front nemůže doručit zprávu, vloží ji do fronty nedoručených zpráv (nedoručená zpráva). Přidá do zprávy záhlaví (obsahující více informací o řízení) a změní toto pole na *Format* .

Pro správce front existuje několik *vestavěných formátů* s názvy začínajícími MQ, například MQFMT_STRING. Pokud tyto požadavky nesplňují vaše potřeby, můžete definovat vlastní formáty (*uživatelem definované formáty*), ale nesmíte používat názvy začínající řetězcem MQ pro tyto účely.

Když vytváříte a používáte vlastní formáty, musíte napsat uživatelskou proceduru pro převod dat, která bude podporovat program získávajícího zprávu pomocí příkazu MQGMO_CONVERT.

CodedCharSetId
Definuje znakovou sadu znakových dat ve zprávě. Chcete-li nastavit tuto znakovou sadu na hodnotu správce front, můžete toto pole nastavit na konstantu MQCCSI_Q_MGR nebo MQCCSI_INHERIT.

Když obdržíte zprávu z fronty, porovnejte hodnotu pole *CodedCharSetId* s hodnotou, kterou vaše aplikace očekává. Pokud se tyto dvě hodnoty liší, možná budete muset konvertovat jakákoli znaková data ve zprávě nebo použít uživatelskou proceduru pro převod dat, je-li k dispozici.

Encoding
Popisuje formát číselných dat zprávy, která obsahují binární celá čísla, packed-decimal celá čísla a čísla s pohyblivou řádovou čárkou. Typicky je kódován podle konkrétního počítače, na kterém je správce front spuštěn.

Když vložíte zprávu do fronty, v poli *Encoding* obvykle uvedete konstantu MQENC_NATIVE. To znamená, že kódování vašich dat zprávy je stejné jako kódování v počítači, na kterém je aplikace spuštěna.

Když obdržíte zprávu z fronty, porovnejte hodnotu pole *Encoding* v popisovači zprávy s hodnotou konstanty MQENC_NATIVE na vašem počítači. Pokud se tyto dvě hodnoty liší, možná budete muset převést jakákoli numerická data ve zprávě nebo použít uživatelskou proceduru pro převod dat, je-li k dispozici.

Převod dat aplikace

Je možné, že data aplikace bude třeba převést na znakovou sadu a kódování požadované jinou aplikací, kde jsou dotčeny různé platformy.

Lze ji převést na odesílající správce front nebo na přijímajícího správce front. Pokud knihovna vestavěných formátů nevyhovuje vašim potřebám, můžete definovat své vlastní. Typ konverze závisí na formátu zprávy, který je zadán v poli formátu deskriptoru zpráv, MQMD.

Poznámka: Zprávy s uvedeným parametrem MQFMT_NONE nejsou převedeny.

Převod v odesílajícím správci front

Nastavte atribut kanálu CONVERT na hodnotu YES, pokud pro převod dat aplikace potřebujete odesílající agent kanálu zpráv (MCA).

Konverze se provádí na odesílajícím správci front pro určité vestavěné formáty a pro uživatelem definované formáty, je-li dodána vhodná uživatelská procedura.

Formáty pro sestavení

Patří k nim:

- Zprávy, které jsou všechny znaky (s použitím názvu formátu MQFMT_STRING)
- IBM MQ definované zprávy, například Programovatelné příkazové formáty

  Produkt IBM MQ používá zprávy ve formátu Programovatelný příkaz pro zprávy a události administrace (použité jméno formátu je MQFMT_ADMIN v tomto případě). Pro své vlastní zprávy můžete použít stejný formát (s použitím názvu formátu MQFMT_PCF) a využívat výhod vestavěného převodu dat.

Všechny vestavěné formáty správce front mají názvy začínající řetězcem MQFMT. Jsou uvedeny a popsány ve formátu Formát.

Formáty definované aplikací

Pro uživatelem definované formáty musí být převod dat aplikací prováděn ukončovacím programem konverze dat (další informace viz “Zápis uživatelských procedur pro převod dat” na stránce 948). V prostředí typu klient-server se uživatelská procedura načte na server a provede se převod.

Převod v přijímajícím správci front

Data zprávy aplikace mohou být převedena přijímajícím správcem front jak pro vestavěné, tak pro uživatelem definované formáty.

Převod se provádí během zpracování volání MQGET, pokud jste zadali volbu MQGMO_CONVERT. Podrobnosti naleznete v části Volby .

Kódové znakové sady

Produkty IBM MQ podporují kódované znakové sady, které jsou poskytovány základním operačním systémem.

Při vytváření správce front je použit identifikátor kódované znakové sady (CCSID) správce front, který je založen na základním prostředí. Pokud se jedná o smíšenou kódovou stránku, produkt IBM MQ používá SBCS část smíšené kódové stránky jako identifikátor CCSID správce front.

U obecného převodu dat, pokud základní operační systém podporuje kódové stránky DBCS, může jej produkt IBM MQ použít.

Podrobnosti o kókónovaných znakových sadách, které podporuje, najdete v dokumentaci k operačnímu systému.

Při zápisu aplikací, které zasahují do více platforem, je třeba brát v úvahu převod dat aplikací, názvy formátů a uživatelské procedury. Informace o vyvolání a zápisu uživatelských procedur pro převod dat naleznete v příručce “Zápis uživatelských procedur pro převod dat” na stránce 948 .

Priority zpráv

Prioritu zprávy můžete buď nastavit na numerickou hodnotu, nebo nechat zprávu nastavit výchozí prioritu fronty.

Při vkládání zprávy do fronty jste nastavili prioritu zprávy (v poli *Priority* ve struktuře MQMD). Pro prioritu můžete nastavit číselnou hodnotu, nebo můžete nechat zprávu nastavit výchozí prioritu fronty.

Atribut **MsgDeliverySequence** fronty určuje, zda jsou zprávy ve frontě ukládány do posloupnosti FIFO (první dovnitř, první ven) nebo ve FIFO v rámci posloupnosti priority. Je-li tento atribut nastaven na hodnotu MQMDS_PRIORITY, jsou zprávy zařazeny do fronty s prioritou určenou v poli *Priority* jejich deskriptoru zpráv, ale pokud je nastavena na hodnotu MQMDS_FIFO, zprávy jsou zařazeny do fronty s výchozí prioritou fronty. Zprávy se stejnou prioritou se ukládají ve frontě v pořadí příchodu.

Atribut **DefPriority** fronty nastavuje výchozí hodnotu priority pro zprávy vkládané do této fronty. Tato hodnota je nastavena při vytvoření fronty, ale lze ji později změnit. Alias front a lokální definice vzdálených front, mohou mít odlišné výchozí priority ze základních front, na které se tyto názvy řeší. Je-li v cestě rozpoznání více než jedna definice fronty (viz “Rozpoznání názvu” na stránce 715), je výchozí priorita převzata z hodnoty atributu **DefPriority** fronty zadané v otevřeném příkazu (v době operace vložení).

Hodnota atributu **MaxPriority** správce front je maximální prioritou, kterou můžete přiřadit ke zprávě zpracovávané daným správcem front. Hodnotu tohoto atributu nelze změnit. V produktu IBM MQmá atribut hodnotu 9; můžete vytvořit zprávy, které mají priority mezi 0 (nejnižší) a 9 (nejvyšší).

Vlastnosti zprávy

Pomocí vlastností zprávy můžete aplikaci umožnit vybrat zprávy ke zpracování nebo načíst informace o zprávě bez přístupu k záhlavím MQMD nebo MQRFH2 . Usnadňují také komunikaci mezi aplikacemi IBM MQ a JMS .

Vlastnost zprávy jsou data přidružená ke zprávě skládající se z textového názvu a hodnoty určitého typu. Vlastnosti zpráv jsou používány selektory zpráv k filtrování publikací do témat nebo k selektivnímu získání zpráv z front. Vlastnosti zpráv lze použít k zahrnutí obchodních dat nebo informací o stavu bez nutnosti jejich uložení do dat aplikace. Aplikace nemusí přistupovat k datům v deskriptoru MQMD (MQ Message Descriptor) nebo v záhlaví MQRFH2 , protože pole v těchto datových strukturách mohou být přístupná jako vlastnosti zprávy pomocí volání funkce rozhraní MQI (Message Queue Interface).

Použití vlastností zpráv v produktu IBM MQ imituje použití vlastností v produktu JMS. To znamená, že můžete nastavit vlastnosti v aplikaci JMS a načíst je v procedurálním produktu IBM MQ nebo v jiném směru. Chcete-li vlastnost zpřístupnit aplikaci produktu JMS , přiřaďte ji předponou "usr"; ta je poté k dispozici (bez předpony) jako vlastnost uživatele zprávy produktu JMS . Například vlastnost IBM MQ *usr.myproperty* (znakový řetězec) je přístupná pro aplikaci JMS pomocí volání JMS `message.getStringProperty('myproperty')`. Všimněte si, že aplikace JMS nemohou přistoupit k vlastnostem s předponou "usr", pokud obsahují dvě nebo více U+002E (".") znaků. Vlastnost bez předpony a ne U+002E (".") jsou považovány za předponu "usr", jako by se s předponou "usr". A naopak, lze k uživatelské vlastnosti nastavené v aplikaci JMS přistupovat v aplikaci IBM MQ přidáním "usr". zadejte předponu názvu vlastnosti, která se bude provádět při volání MQINQMP.

Vlastnosti zprávy a délka zprávy

Použijte atribut správce front *MaxPropertiesLength* k řízení velikosti vlastností, které mohou tékat se všemi zprávami ve správci front IBM MQ .

Obecně platí, že když pomocí příkazu MQSETMP nastavíte vlastnosti, velikost vlastnosti je délka názvu vlastnosti v bajtech, plus délka hodnoty vlastnosti v bajtech, jak byla předána do volání funkce MQSETMP.

Je možné, že znaková sada názvu vlastnosti a hodnota vlastnosti se změní během přenosu zprávy na místo určení, protože je lze převést na Unicode; v tomto případě by se velikost vlastnosti mohla změnit.

Ve volání MQPUT nebo MQPUT1 se vlastnosti zprávy nepočítají směrem k délce zprávy pro frontu a správce front, ale počítají se směrem k délce vlastností, jak je vnímá správce front (ať už byly nastaveny pomocí volání MQI pro vlastnost zprávy nebo nikoli).

Pokud velikost vlastností překračuje maximální délku vlastností, je zpráva odmítnuta s parametrem MQRC_PROPERTIES_TOO_BIG. Vzhledem k tomu, že velikost vlastností je závislá na její reprezentaci, je třeba nastavit maximální délku vlastností na hrubou úroveň.

Je možné, že aplikace úspěšně vloží zprávu s vyrovnávací pamětí, která je větší než hodnota *MaxMsgLength*, pokud vyrovnávací paměť obsahuje vlastnosti. Je tomu tak proto, že i když jsou představovány jako prvky MQRFH2 , vlastnosti zprávy se nepočítá směrem k délce zprávy. Pole záhlaví MQRFH2 se přidávají k délce vlastností pouze v případě, že je obsažena jedna nebo více složek a každá složka v záhlaví obsahuje vlastnosti. Pokud se jedna nebo více složek nachází v záhlaví MQRFH2 a žádná složka neobsahuje vlastnosti, bude místo toho počet polí záhlaví MQRFH2 započítává směrem k délce zprávy.

Ve volání MQGET se vlastnosti zprávy nepočítají směrem k délce zprávy, pokud jde o frontu a správce front. Protože se však vlastnosti počítají samostatně, je možné, že vyrovnávací paměť vrácená voláním MQGET je větší než hodnota atributu *MaxMsgLength* .

Nemějte na paměti dotaz na hodnotu parametru *MaxMsgLength* a pak přidělte vyrovnávací paměť této velikosti před voláním MQGET; místo toho alokujte vyrovnávací paměť, kterou považujete za dostatečně velkou. Pokud příkaz MQGET selže, alokujte vyrovnávací paměť vedenou velikostí parametru *DataLength* .

Parametr *DataLength* volání MQGET vrací délku v bajtech dat aplikace a všechny vlastnosti vrácené ve vyrovnávací paměti, které jste zadali, pokud v struktuře MQGMO není zadán popisovač zprávy.

Parametr *Buffer* volání MQPUT obsahuje data zprávy aplikace, která mají být odeslána, a všechny vlastnosti reprezentované v datech zprávy.

Při toku do správce front, který je starší než IBM WebSphere MQ 7.0, vlastnosti zprávy, kromě vlastností v deskriptoru zpráv, se počítají do délky zprávy. Proto byste měli buď zvýšit hodnotu atributu *MaxMsgLength* kanálů, které se chystáte na systém dříve, než IBM WebSphere MQ 7.0 , aby vykompenzovali skutečnost, že pro každou zprávu může být odeslána více dat. Případně můžete snížit délku fronty nebo správce front *MaxMsgLength*, aby byla celková úroveň dat posílaná po celém systému stejná.

Pro vlastnosti zprávy je k dispozici omezení délky 100 MB, kromě deskriptoru zprávy nebo přípony pro každou zprávu.

Velikost vlastnosti ve vnitřní reprezentaci je délka názvu, plus velikost její hodnoty, plus některé řídicí údaje pro vlastnost. K dispozici jsou také některá řídicí data pro sadu vlastností po přidání jedné vlastnosti do zprávy.

Názvy vlastností

Název vlastnosti je znakový řetězec. Některá omezení se vztahují na jeho délku a sadu znaků, které lze použít.

Název vlastnosti je znakový řetězec rozlišující velikost písmen, omezen na +4095 znaků, pokud není jinak omezen kontextem. Tento limit je obsažen v konstantě MQ_MAX_PROPERTY_NAME_LENGTH.

Překročíte-li tuto maximální délku při použití volání MQI pro vlastnost zprávy, volání selže s kódem příčiny MQRC_PROPERTY_NAME_LENGTH_ERR.

Protože v produktu JMSneexistuje maximální délka názvu vlastnosti, je možné, aby aplikace JMS nastavila platný název vlastnosti produktu JMS , který není platným názvem vlastnosti produktu IBM MQ , je-li uložen ve struktuře MQRFH2 .

V tomto případě jsou při analýze použity pouze prvních 4095 znaků názvu vlastnosti; následující znaky jsou ořízntuty. To by mohlo způsobit, že aplikace používá selektory, aby se shodovala s výběrovým řetězcem, nebo aby odpovídala řetězci, když se neočekává, protože více než jedna vlastnost by mohla

oříznout na stejný název. Je-li název vlastnosti zkrácen, produkt WebSphereMQ vydá zprávu s chybovou zprávou.

Všechny názvy vlastností musí odpovídat pravidlům definovaným specifikací jazyka Java pro identifikátory Java , s výjimkou toho, že znak Unicode U+002E (.) je povolen jako součást názvu-ale ne pro začátek. Pravidla pro identifikátory Java se rovnají těm, které jsou obsaženy ve specifikaci JMS pro názvy vlastností.

Operátory mezer a operátorů porovnání jsou zakázány. Vložené hodnoty null jsou povoleny v názvu vlastnosti, ale nejsou doporučeny. Pokud použijete vložené hodnoty null, zabráníte použití konstanty MQVS_NULL_TERMINATED při použití s strukturou MQCHARV k zadání řetězců s proměnnou délkou.

Uchovat názvy vlastností jsou jednoduché, protože aplikace mohou vybírat zprávy založené na názvech vlastností a převod mezi znakovou sadou názvu a selektorem může způsobit neočekávané selhání výběru.

Názvy vlastností IBM MQ používají znak U+002E (.) pro logické seskupení vlastností. Tím se rozdělí obor názvů pro vlastnosti. Vlastnosti s následujícími předponami, ve všech směnách malých nebo velkých písmen jsou vyhrazeny pro použití produktem:

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

Dobrým způsobem, jak zabránit kolizi názvů, je zajistit, aby všechny aplikace předčíslovaly své vlastnosti zprávy svým názvem internetové domény. Pokud například vyvíjíte aplikaci s použitím názvu domény `ourcompany.com` , můžete pojmenovat všechny vlastnosti s předponou `com.ourcompany`. Tato konvence pojmenování také umožňuje snadný výběr vlastností; aplikace se může například dotázat na všechny vlastnosti zprávy, počínaje produktem `com.ourcompany.%`.

Další informace o použití názvů vlastností naleznete v tématu Omezení názvu vlastnosti .

Omezení názvu vlastnosti
Když pojmenujete vlastnost, musíte dodržovat určitá pravidla.

Na názvy vlastností se vztahují následující omezení:

1. Vlastnost nesmí začínat následujícími řetězci:
   - "JMS"-vyhrazeno pro použití produktem IBM MQ classes for JMS.
   - "usr.JMS"-není platné.

   Jediné výjimky jsou následující vlastnosti poskytující synonyma pro vlastnosti produktu JMS :

Vlastnost	**Synonymum**
JMSCorrelationID	Kořen.MQMD.CorrelId nebo jms.Cid
JMSDeliveryMode	Kořen.MQMD.Persistence nebo jms.Dlv
JMSDestination	jms.Dst
JMSExpiration	Kořen.MQMD.Expiry nebo jms.Exp
JMSMessageID	Kořen.MQMD.MsgId
JMSPriority.	Kořen.MQMD.Priority nebo jms.Pri

Vlastnost	Synonymum
JMSRedelivered	Kořen.MQMD.BackoutCount
JMSReplyTo (řetězec kódovaný jako identifikátor URI)	Kořen.MQMD.ReplyToQ nebo Kořen.MQMD.ReplyToQMgr nebo jms.Rto
JMSTimestamp	Kořen.MQMD.PutDate nebo Kořen.MQMD.PutTime nebo jms.Tms
JMSType.	mcd.Type nebo mcd.Set nebo mcd.Fmt
JMSXAppID	Kořen.MQMD.PutApplName .
JMSXDeliveryCount	Kořen.MQMD.BackoutCount
JMSXGroupID	Kořen.MQMD.GroupId nebo jms.Gid
JMSXGroupSeq	Kořen.MQMD.MsgSeqNumber nebo jms.Seq
JMSXUserID	Kořen.MQMD.UserIdentifier

Tato synonyma umožňují aplikacím MQI přistupovat k vlastnostem produktu JMS podobným způsobem jako aplikaci klienta IBM MQ classes for JMS . Z těchto vlastností lze pomocí rozhraní MQI nastavit pouze JMSCorrelationID, JMSReplyTo, JMSTType, JMSXGroupIDa JMSXGroupSeq .

Všimněte si, že vlastnosti JMS_IBM_ *, které jsou k dispozici v rámci produktu IBM MQ classes for JMS , nejsou k dispozici s použitím rozhraní MQI. Pole, ke kterým lze přistupovat k referenci vlastností JMS_IBM_ *, lze v aplikacích MQI používat i jinými způsoby.

2. Vlastnost nesmí být volána ani v žádné směsi malých nebo velkých, "NULL", "TRUE", "FALSE", "NOT", "AND", "OR", "BETWEEN", "LIKE", "IN", "IS" a "ESCAPE". Jedná se o názvy klíčových slov SQL použitých ve výběrových řetězcích.
3. Název vlastnosti začínající " mq " v libovolné kombinaci malých a velkých písmen a ne začínající "mq_usr" může obsahovat pouze jeden "." character (U+002E). Vícenásobné "." znaky nejsou ve vlastnostech s těmito předponami povoleny.
4. Dva "." Znaky musí obsahovat jiné znaky mezi; nemůžete mít prázdný bod v hierarchii. Podobně název vlastnosti nesmí končit znakem "." Znak.
5. Pokud aplikace nastaví vlastnost "a.b" a pak vlastnost "a.b.c", není jasné, zda v hierarchii "b" obsahuje hodnotu nebo jiné logické seskupení. Taková hierarchie je "smíšený obsah" a tato položka není podporována. Nastavení vlastnosti, která způsobí smíšený obsah, není povoleno.

Tato omezení jsou vynucována mechanismem ověřování následujícím způsobem:

- Názvy vlastností jsou ověřovány při nastavení vlastnosti pomocí volání funkce MQSETMP-nastavení vlastnosti zprávy , pokud bylo při vytvoření popisovače zprávy požadováno ověření. Je-li proveden pokus o ověření platnosti vlastnosti a v důsledku chyby ve specifikaci názvu vlastnosti došlo k selhání, kód dokončení je MQCC_FAILED s důvodem:
  - Objekt MQRC_PROPERTY_NAME_ERROR z důvodů 1-4.
  - Objekt MQRC_MIXED_CONTENT_NOT_ALLOWED z důvodu 5.
- Názvy vlastností, které jsou zadány přímo jako prvky MQRFH2 , nejsou garantovány, aby byly ověřeny voláním MQPUT.

Pole deskriptoru zpráv jako vlastnosti
Většina polí deskriptoru zprávy může být považována za vlastnosti. Název vlastnosti je sestaven přidáním předpony do názvu pole deskriptoru zpráv.

Pokud chce aplikace MQI identifikovat vlastnost zprávy obsaženou v poli deskriptoru zpráv, například v řetězci selektoru nebo pomocí rozhraní API vlastností zprávy, použijte následující syntaxi:

Název vlastnosti	Pole deskriptoru zpráv
Root.MQMD.*Pole*	*Pole*

Zadejte *Field* se stejným případem jako pro pole struktury MQMD v deklaraci jazyka C. Název vlastnosti `Root.MQMD.AccountingToken` například přistupuje k poli `AccountingToken` v deskriptoru zprávy.

Pole `StrucId` a `Version` deskriptoru zpráv nejsou přístupná pomocí zobrazené syntaxe.

Pole deskriptoru zpráv nejsou nikdy reprezentována v záhlaví MQRFH2 jako pro jiné vlastnosti.

Pokud data zprávy začínají řetězcem MQMDE, který je uznán správcem front, lze k polím MQMDE přistupovat s použitím popsané notace `Root.MQMD.`*`Field`* . V tomto případě jsou pole MQMDE považována za logickou část MQMD z perspektivy vlastností. Viz Přehled prostředí MQMDE.

Typy a hodnoty dat vlastností

Vlastnost může být logický, bajtový řetězec, znakový řetězec, nebo číslo s pohyblivou řádovou čárkou nebo celé číslo. Vlastnost může uložit jakoukoli platnou hodnotu v rozsahu datového typu, pokud není jinak omezen kontextem.

Datový typ hodnoty vlastnosti musí být jedna z následujících hodnot:

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

Vlastnost může existovat, ale nemá definovanou hodnotu; jedná se o vlastnost s hodnotou null. Vlastnost s hodnotou Null se liší od vlastnosti bajtu (MQBYTE []) nebo vlastnosti znakového řetězce (MQCHAR []) v tom, že má definovanou, ale prázdnou hodnotu, tj. jednu s hodnotou s nulovou délkou.

Řetězec bajtů není platným datovým typem vlastnosti v produktu JMS nebo XMS. Je doporučeno nepoužívat vlastnosti řetězce bajtů ve složce *usr* .

Výběr zpráv z front

Zprávy z front můžete vybrat pomocí polí MsgId a CorrelId na volání MQGET nebo pomocí řetězce SelectionString na volání MQOPEN nebo MQSUB.

Selektory.

Selektor zpráv je řetězec s proměnnou délkou, který aplikace používá k registraci svého zájmu pouze v těch zprávách, které mají vlastnosti, které vyhovují dotazu SQL (Structured Query Language), který představuje řetězec výběru.

Výběr pomocí funkčních volání MQSUB a MQOPEN

Příkaz *`SelectionString`*, který je strukturou typu MQCHARV, je použit k provedení výběru pomocí volání MQSUB a MQOPEN.

Struktura *`SelectionString`* se používá k předání řetězce výběru s proměnnou délkou do správce front.

Identifikátor CCSID přidružený k řetězci selektoru se nastavuje prostřednictvím pole VSCCSID struktury MQCHARV. Použitá hodnota musí být CCSID, který je podporován pro řetězce selektoru. Viz Převod kódových stránek , kde najdete seznam podporovaných kódových stránek.

Zadání CCSID, pro který neexistuje žádná podporovaná IBM MQ konverze Unicode, má za následek chybu MQRC_SOURCE_CCSID_ERROR. Tato chyba je vrácena v době, kdy je selektor představen správci front, který je spuštěn ve volání MQSUB, MQOPEN nebo MQPUT1 .

Výchozí hodnota pro pole *VSCCSID* je MQCCSI_APPL, která označuje, že se CCSID výběrového řetězce shoduje s CCSID správce front nebo CCSID klienta, pokud je připojen přes klienta. Konstanta MQCCSI_APPL může být však potlačena aplikací předefinováním před kompilací.

Pokud selektor MQCHARV představuje řetězec s hodnotou NULL, nebude proveden žádný výběr pro spotřebitele zpráv a zprávy jsou doručeny, jako by se nepoužil selektor.

Maximální délka řetězce výběru je omezena pouze tím, co může být popsáno v poli MQCHARV *VSLength*.

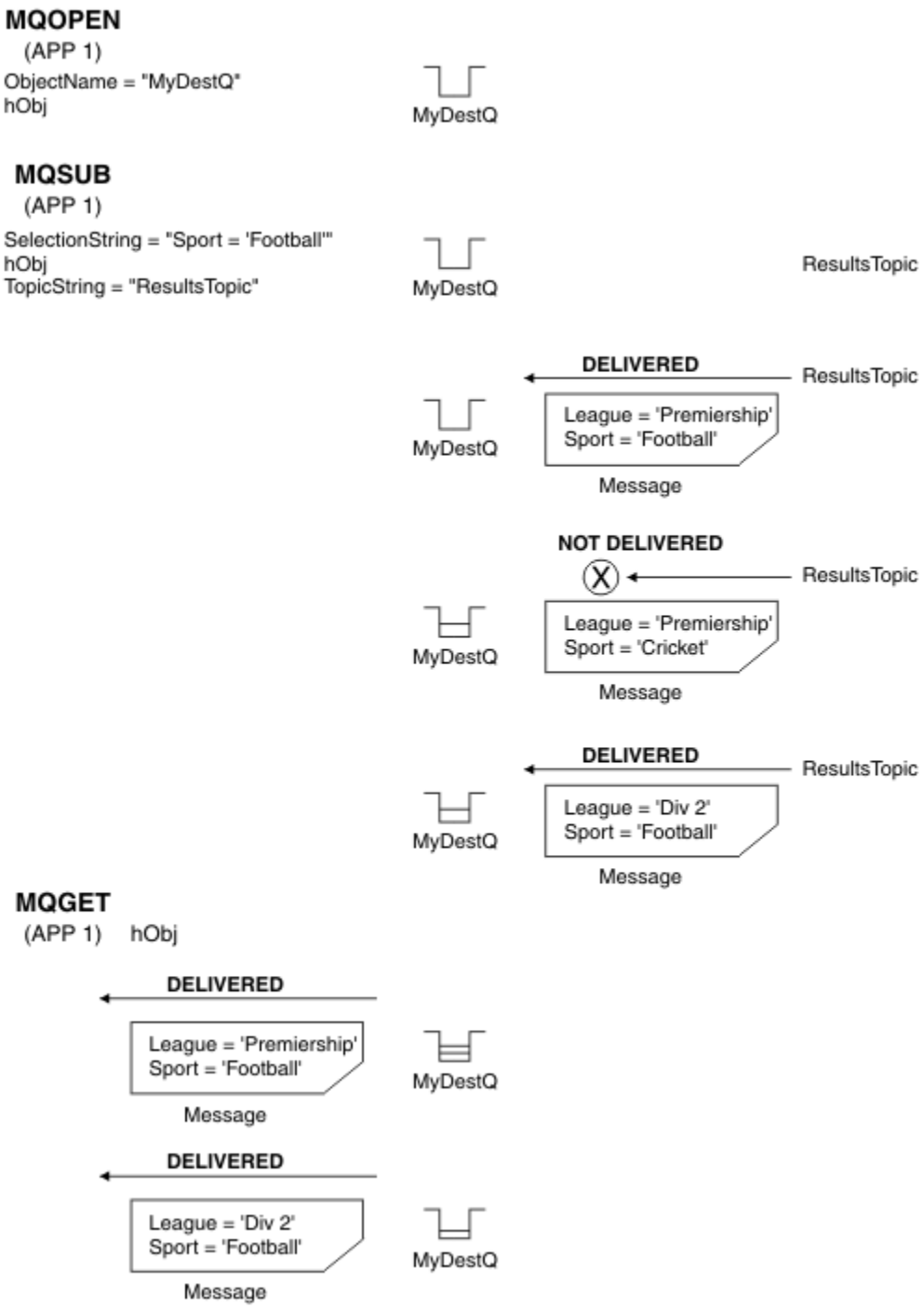
Hodnota SelectionString je vrácena ve výstupu z volání MQSUB s použitím volby MQSO_RESUME, pokud jste poskytli vyrovnávací paměť a v parametru VSBufSizeje kladná délka vyrovnávací paměti. Pokud vyrovnávací paměť nezadáte, vrátí se v poli VSLLength pole MQCHARV pouze délka řetězce výběru. Je-li poskytnutá vyrovnávací paměť menší než prostor potřebný k navrácení pole, vrátí se ve vyrovnávací paměti pouze bajty VSBufSize .

Aplikace nemůže změnit řetězec výběru, aniž by nejprve zavíral popisovač do fronty (pro MQOPEN) nebo odběr (pro MQSUB). Poté lze zadat nový řetězec výběru v rámci následné volání MQOPEN nebo MQSUB.

MQOPEN
Pomocí funkce MQCLOSE zavřete otevřený popisovač a poté zadejte nový řetězec výběru v rámci následného volání MQOPEN.

MQSUB
Pomocí příkazu MQCLOSE zavřete vrácený popisovač odběru (hSub), poté zadejte nový řetězec výběru při následném volání MQSUB.

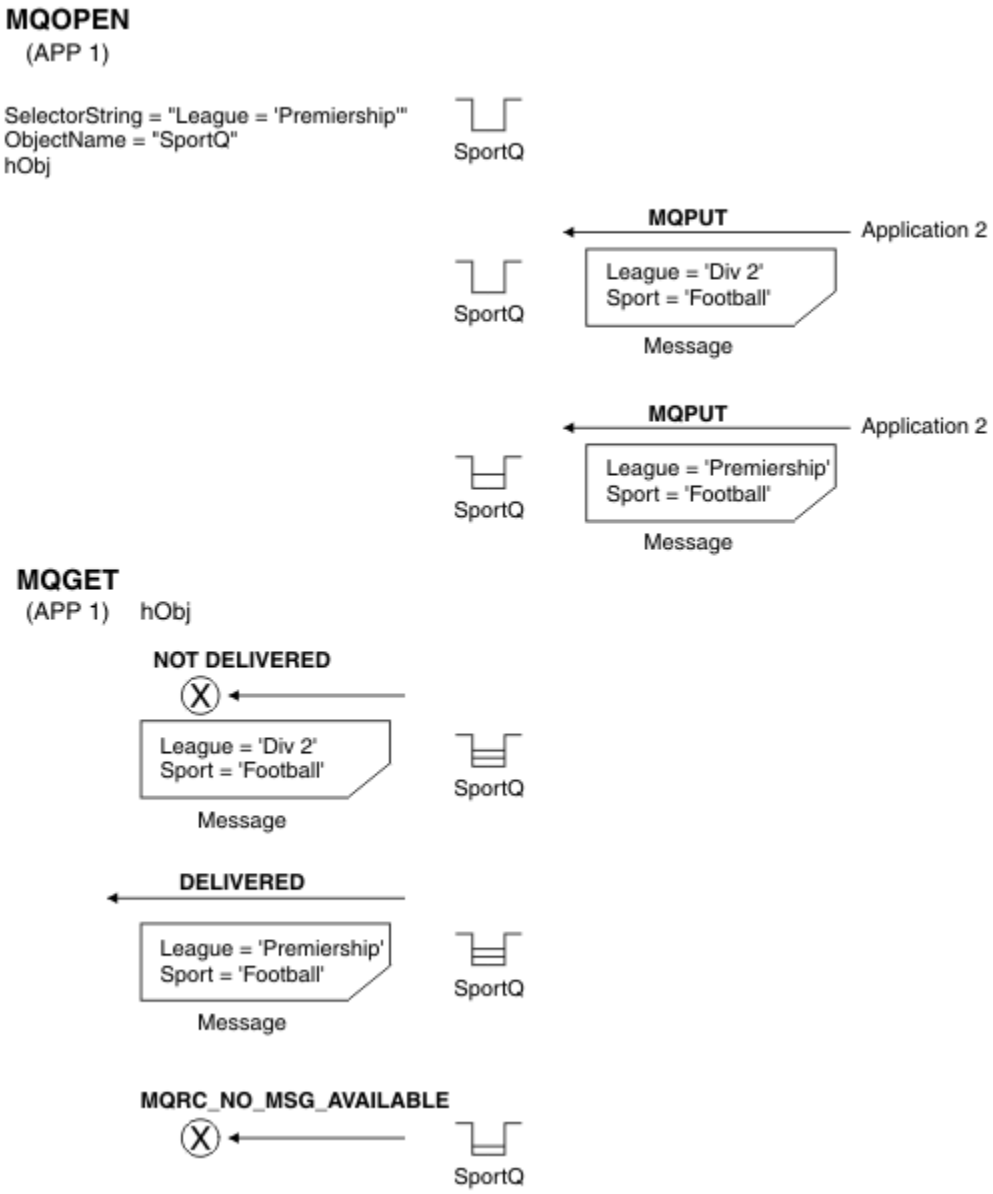
Produkt Obrázek 3 na stránce 27 zobrazuje proces výběru pomocí volání MQSUB.



Obrázek 3. Výběr pomocí volání MQSUB

Pomocí pole *SelectionString* ve struktuře MQSD lze do volání MQSUB předat selektor voláním MQSUB. Výsledkem předání v selektoru MQSUB je, že jsou k dispozici pouze zprávy publikované v rámci daného tématu, které odpovídají zadanému výběrovému řetězci, k dispozici v cílové frontě.

Produkt Obrázek 4 na stránce 28 zobrazuje proces výběru pomocí volání MQOPEN.



Obrázek 4. Výběr pomocí volání MQOPEN

Pomocí pole *SelectionString* ve struktuře MQOD lze k volání MQOPEN předat selektor. Výsledkem předání selektoru v rámci volání MQOPEN je to, že se spotřebiteli zpráv doručí pouze zprávy v otevřené frontě, které odpovídají selektoru.

Hlavní použití pro selektor na volání MQOPEN je pro případ dvoubodového spojení, kdy se aplikace může rozhodnout přijímat pouze ty zprávy ve frontě, které odpovídají selektoru. Předchozí příklad ukazuje jednoduchý scénář, kdy dvě zprávy jsou vloženy do fronty otevřené MQOPEN, ale aplikace ji přijme pouze jedna, protože to je jediná, která se shoduje se selektorem.

Všimněte si, že následující volání MQGET má za následek MQRC_NO_MSG_AVAILABLE, protože ve frontě neexistují žádné další zprávy, které odpovídají danému selektoru.

Související pojmy

"Pravidla řetězce výběru a omezení" na stránce 35
Seznamte se s těmito pravidly o tom, jak jsou řetězce výběru interpretovány a znaková omezení, abyste se vyhnuli možným problémům při použití selektorů.

Chování výběru

Přehled chování výběru produktu IBM MQ .

Pole ve struktuře MQMDE jsou považována za vlastnosti zprávy pro odpovídající vlastnosti deskriptoru zpráv, pokud MQMD:

- Má formát MQFMT_MD_EXTENSION
- Je okamžitě následováno platnou strukturou MQMDE
- Je verze jedna nebo obsahuje pouze dvě pole s výchozí verzí.

Je možné, aby řetězec výběru byl vyhodnocen buď jako TRUE, nebo FALSE před tím, než dojde ke shodě s vlastnostmi zprávy. Může tomu tak být například v případě, že je řetězec výběru nastaven na hodnotu `"TRUE <> FALSE"`. Toto včasné vyhodnocení je zaručeno pouze v případě, že v řetězci výběru nejsou žádné odkazy na vlastnosti zprávy.

Je-li řetězec výběru před všemi vlastnostmi zprávy interpretován jako TRUE, budou doručeny všechny zprávy publikované na téma přihlášené odběratelem. Je-li řetězec výběru před zvažováním vlastností zprávy vyhodnocen na hodnotu FALSE, bude vrácen kód příčiny MQRC_SELECTOR_ALWAYS_FALSE a kód dokončení MQCC_FAILED u volání funkce, které je představeno selektorem.

I když zpráva neobsahuje žádné vlastnosti zprávy (jiné než vlastnosti záhlaví), pak může být stále vhodný pro výběr. Pokud se řetězec výběru odkazuje na vlastnost zprávy, která neexistuje, předpokládá se, že tato vlastnost má hodnotu NULL nebo 'Unknown'.

Zpráva může například stále splňovat výběrový řetězec, například `'Color IS NULL'`, kde `'Color'` neexistuje jako vlastnost zprávy ve zprávě.

Výběr lze provést pouze u vlastností, které jsou přidruženy ke zprávě, nikoli ke zprávě samotné, pokud není k dispozici poskytovatel rozšířeného výběru zpráv. Výběr lze na informačním obsahu zprávy provést pouze tehdy, je-li k dispozici rozšířený poskytovatel výběru zpráv.

Ke každé vlastnosti zprávy je přidružen typ. Když provádíte výběr, musíte se ujistit, že hodnoty použité ve výrazech na vlastnosti testovací zprávy mají správný typ. Pokud dojde k neshodě typu, bude daný výraz vyhodnocen jako FALSE.

Je vaší odpovědností zajistit, aby výběrový řetězec a vlastnosti zpráv používaly kompatibilní typy.

Výběrová kritéria se i nadále používají na účet neaktivních trvalých odběratelů, takže jsou zachovány pouze zprávy, které odpovídají původně dodanému řetězci výběru.

Výběrové řetězce jsou nealterovatelné, když je trvalý odběr obnoven se změnou (MQSO_ALTER). Je-li při obnovení aktivity trvalého odběratele zobrazen jiný výběrový řetězec, vrátí se aplikaci MQRC_SELECTOR_NOT_ALTERABLE.

Aplikace obdrží návratový kód MQRC_NO_MSG_AVAILABLE, pokud ve frontě není žádná zpráva, která splňuje kritéria výběru.

Pokud aplikace zadá řetězec výběru obsahující hodnoty vlastností, jsou vhodné pro výběr pouze ty zprávy, které obsahují odpovídající vlastnosti. Odběratel například určuje řetězec výběru "a = 3" a je publikována zpráva, která neobsahuje žádné vlastnosti, nebo vlastnosti, kde 'a' neexistuje, nebo se nerovná 3. Odběratel tuto zprávu neobdrží do své cílové fronty.

Výkon systému zpráv

Výběr zpráv z fronty vyžaduje, aby produkt IBM MQ postupně kontrolovaly každou zprávu ve frontě. Zprávy jsou zkontrolovány, dokud není nalezena zpráva, která odpovídá kritériím výběru, nebo nejsou k dispozici žádné další zprávy ke kontrole. Proto se výkon systému zpráv utrpí, pokud je v hlubokých frontách použit výběr zpráv.

Chcete-li optimalizovat výběr zpráv v hlubokých frontách, je-li výběr založen na JMSCorrelationID nebo JMSMessageID, použijte řetězec výběru ve tvaru:

- JMSCorrelationID = 'ID:*correrelation_id*'
- JMSMessageID= 'ID:*message_id*'

kde:

- *correrelation_id* je řetězec obsahující standardní korelační identifikátor IBM MQ .
- *message_id* je řetězec obsahující standardní identifikátor zprávy IBM MQ .

Poznámka: Selektor by měl odkazovat pouze na jednu z vlastností. Použití selektoru, který má jeden z těchto formátů, nabízí významné zlepšení výkonu při výběru volby JMSCorrelationID a nabízí mezní zlepšení výkonu pro JMSMessageID. Další informace viz “Selektory zpráv v JMS” na stránce 122.

Použití komplexních selektorů

Selektory mohou obsahovat mnoho komponent, například:

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

Použití takových komplexních selektorů může mít vážný dopad na výkon a nadměrné požadavky na prostředky. Produkt IBM MQ tak bude chránit systém tím, že nebude zpracovávat příliš složité selektory, které by mohly vést k nedostatku systémových prostředků. K ochraně může dojít u výběrových řetězců, které obsahují více než 100 testů, nebo když produkt IBM MQ zjistí, že se blíží mezní hodnota velikosti zásobníku operačního systému. Měli byste důkladně vyzkoušet a otestovat použití výběrových řetězců s mnoha komponentami na příslušných platformách, abyste se ujistili, že limity ochrany nejsou dosaženy.

Výkon a složitost selektorů lze vylepšit jejich zjednodušením použitím dalších závorek pro sloučení komponent. Příklad:

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

Související pojmy

“Pravidla řetězce výběru a omezení” na stránce 35
Seznamte se s těmito pravidly o tom, jak jsou řetězce výběru interpretovány a znaková omezení, abyste se vyhnuli možným problémům při použití selektorů.

Syntaxe selektoru zpráv

Selektor zpráv produktu IBM MQ je řetězec se syntaxí, která je založena na dílčí sadě syntaxe podmíněného výrazu SQL92 .

Pořadí, ve kterém je vyhodnocován selektor zpráv, je zleva doprava v rámci úrovně priority. Chcete-li toto pořadí změnit, můžete použít závorky. Literály předdefinovaných selektorů a názvy operátorů jsou zapsány zde velkými písmeny; avšak nerozlišují se malá a velká písmena.

Je-li selektor poskytnut přes API, IBM MQ ověřuje syntaktickou správnost selektoru zpráv v době, kdy je prezentován. Je-li syntaxe řetězce výběru nesprávná nebo název vlastnosti není platný a poskytovatel rozšířeného výběru zpráv není k dispozici, vrátí se aplikaci MQRC_SELECTION_NOT_AVAILABLE . Je-li syntaxe výběrového řetězce nesprávná nebo název vlastnosti není platný při obnovení odběru, vrátí se aplikace do aplikace MQRC_SELECTOR_SYNTAX_ERROR . Pokud bylo ověření názvu vlastnosti zakázáno, když byla nastavena vlastnost (nastavením MQCMHO_NONE namísto MQCMHO_VALIDATE) a aplikace následně vloží zprávu s neplatným názvem vlastnosti, tato zpráva se nikdy nevybere.

Když produkt IBM MQ zjistí, že administrativně definovaný selektor odběru používá administrativně definovaný selektor zpráv s použitím parametru **DISPLAY SUB SELTYPE** s hodnotou EXTENDED, není vrácena žádná chyba v čase, kdy je selektor předložen. V tomto případě je kontrola syntaxe výběrového řetězce odložena do doby publikování (viz MQRC_SELECTION_NOT_AVAILABLE).

Selektor může obsahovat:

- Literály:
  - Řetězcové literály jsou uzavřeny v jednoduchých uvozovkách. Dvě po sobě jdoucí jednoduché uvozovky představují jednoduché uvozovky. Příklady jsou 'literal' a 'literal''s '. Jako řetězcové literály Java používají kódování znaků Unicode kódování znaků Unicode. Dvojité uvozovky nelze použít k uzavření řetězcového literálu. Libovolnou posloupnost bajtů lze použít mezi jednoduchými uvozovkami.

– Bajtový řetězec je jeden nebo více dvojic hexadecimálních znaků uzavřených do uvozovek a s předponou `0x`. Příklady jsou `"0x2F1C"` nebo `"0XD43A"`. Délka bajtového řetězce musí být alespoň jeden bajt. Pokud je řetězec bajtů selektoru porovnán se vlastností zprávy typu `MQTYPE_BYTE_STRING`, neprovedou se žádné speciální akce na úvodní nebo koncové nule. S bajty se zachází jako s jiným znakem. Endianness se také nezvažuje. Délka obou bajtových řetězců selektoru i vlastností musí být stejná, a posloupnost bajtů musí být stejná.

  Příklady výběru bajtových řetězců (předpokládá se *myBytes* = `0AFC23`), které se shodují:

  - `"myBytes = "0x0AFC23"" = TRUE`

  Následující výběry řetězců se neshodují:

  - `"myBytes = "0xAFC23"" = MQRC_SELECTOR_SYNTAX_ERROR` (protože počet bajtů není násobkem dvou)
  - `"myBytes = "0x0AFC2300"" = FALSE` (protože koncová nula je významná v porovnání)
  - `"myBytes = "0x000AFC23"" = FALSE` (protože vedoucí nula je významná v porovnání)
  - `"myBytes = "0x23FC0A"" = FALSE` (protože endianness se neuvažuje)

– Hexadecimální čísla začínají nulou a následují velkými nebo malými písmeny `x`. Zbytek literálu obsahuje jeden nebo více platných hexadecimálních znaků. Příklady: `0xA`, `0xAF`, `0X2020`.

– Počáteční nula následovaná jednou nebo více číslicemi v rozsahu 0-7 je vždy interpretována jako začátek oktalového čísla. Nemůže představovat desítkové číslo s předponou nula, například `09` , vrací chybu syntaxe, protože číslo 9 není platnou oktalovou číslicí. Příklady oktalových čísel jsou `0177`, `0713`.

– Přesný číselný literál je číselná hodnota bez desetinné čárky, jako například 57, `-957a` `+62`. Přesný číselný literál může mít koncový znak `L` nebo malá písmena `L`; to nemá vliv na to, jak je číslo uloženo nebo interpretováno. IBM MQ podporuje přesné číslice v rozsahu `-9,223,372,036,854,775,808` až `9,223,372,036,854,775,807`.

– Přibližný numerický literál je číselná hodnota ve vědecké notaci, jako například `7E3` nebo `-57.9E2`, nebo číselná hodnota s desítkovým číslem, například `7.`, `-95.7`nebo `+6.2`. IBM MQ podporuje čísla v rozsahu `-1.797693134862315E+308` až `1.797693134862315E+308`.

  Tento parametr by měl následovat nepovinný znak znaménka (`+` nebo `-`). The significand should be either an integer or a fraction. Zlomková část hodnoty mantise a nemusí mít vedoucí číslici.

  Písmeno nebo malé písmeno `E` označuje začátek volitelného exponentu. Exponent má dekadický radix a číslo části exponentu může být prefixem nepovinným znaménkem.

  Přibližné numerické literály mohou být ukončeny znakem `F` nebo `D` (nerozlišujícím velikost písmen). Tato syntaxe existuje pro podporu metody křížového jazyka značek na jedno nebo dvojitá přesnost čísel. Tyto znaky jsou volitelné a nemají vliv na to, jak je uložen nebo zpracován přibližný numerický literál. Tato čísla jsou vždy uložena a zpracována s použitím dvojité přesnosti.

– Booleovské literály `TRUE` a `FALSE`.

Poznámka: Nekonečná znázornění `IEEE-754` , jako například `NaN`, `+Infinity`, `-Infinity` , nejsou ve výběrových řetězcích podporována. Proto není možné použít tyto hodnoty jako operandy ve výrazu. Negativní nula je zacházeno stejně jako pozitivní nula pro matematické operace.

- Identifikátory:

  Identifikátor je posloupnost znaků s proměnnou délkou, která musí začínat platným znakem začátku identifikátoru následován nulou nebo více platnými znaky identifikátoru. Pravidla pro názvy identifikátorů jsou stejná jako pravidla pro názvy vlastností zpráv, viz “Názvy vlastností” na stránce 22 a “Omezení názvu vlastnosti” na stránce 23 , kde získáte další informace.

  Poznámka: Výběr lze na informačním obsahu zprávy provést pouze tehdy, je-li k dispozici rozšířený poskytovatel výběru zpráv.

  Identifikátory jsou buď odkazy na pole záhlaví, nebo odkazy na vlastnosti. Typ hodnoty vlastnosti v selektoru zpráv musí odpovídat typu použitému k nastavení vlastnosti, ačkoli je to možné, je-li to

možné, je provedena číselná povýšení. Dojde-li k neshodě typů, pak je výsledkem výrazu FALSE. Pokud na vlastnost, která ve zprávě neexistuje, se odkazuje, její hodnota je NULL.

Převody typu, které platí pro metody get pro vlastnosti get, se nepoužijí, když se vlastnost používá ve výrazu selektoru zpráv. Pokud například nastavíte vlastnost jako řetězcovou hodnotu a pak použijete selektor k zadání dotazu na číselnou hodnotu, vrátí výraz FALSE.

Pole JMS a názvy vlastností, které jsou mapovány na názvy vlastností nebo názvy polí MQMD , jsou platnými identifikátory v řetězci výběru. Produkt IBM MQ mapuje rozpoznané pole JMS a názvy vlastností na hodnoty vlastností zprávy. Další informace viz “Selektory zpráv v JMS” na stránce 122. Jako příklad, řetězec výběru `"JMSPriority >="` se vybere ve vlastnosti `Pri` ve složce `jms` aktuální zprávy.

- Přetečení/podtečení:

  Pro desetinné i přibližné číselné hodnoty nejsou definovány následující podmínky:

  – Určení čísla, které je mimo definovaný rozsah
  – Určení aritmetického výrazu, který by způsobil přetečení nebo podtečení

  Pro tyto podmínky nejsou provedeny žádné kontroly.

- Netisknutelné:

  Definováno jako mezera, posun o znak, nový řádek, návrat vozíku, vodorovný tabelátor nebo vertikální tabulátor. Následující znaky Unicode jsou rozpoznány jako bílé znaky:

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

- Výrazy:

  – Selektor je podmíněný výraz. Selektor, který se vyhodnocuje na skutečné shody; selektor, který se vyhodnocuje na hodnotu false nebo neznámý, se neshoduje.
  – Aritmetické výrazy se skládají z sebe, aritmetických operací, identifikátorů (hodnota identifikátoru je považována za číselný literál) a numerické literály.
  – Podmíněné výrazy se skládají z vlastních, porovnávacích operací a logických operací.

- Standardní bracketing () pro nastavení pořadí, ve kterém jsou výrazy vyhodnocovány, je podporován.
- Logické operátory v pořadí priority: NOT, AND, OR.
- Operátory porovnání: =, >, >=, <, <=, <> (nerovná se).

  – Dva bajtové řetězce se rovnají, pouze pokud řetězce mají stejnou délku a pořadí bajtů je stejné.
  – Porovnávají se pouze hodnoty stejného typu. Výjimkou je, že je platné porovnat přesné číselné hodnoty a přibližné číselné hodnoty, (požaduje se konverze typu je definována pomocí pravidel Java číselného povýšení). Pokud dojde k pokusu o porovnání různých typů, je selektor vždy nepravdivý.

  - Porovnání řetězců a logických hodnot je omezeno na = a <>. Dva řetězce jsou shodné pouze tehdy, obsahují-li stejnou posloupnost znaků.
- Aritmetické operátory v pořadí priority:
  - +, - unární.
  - Rozdělení * násobení a / .
  - + sčítání a odčítání - .
  - Aritmetické operace na hodnotě NULL nejsou podporovány. Pokud se o tyto pokusy pokusí, úplný selektor je vždy nepravdivý.
  - Aritmetické operace musí používat Java číselnou propagaci.
- arithmetic-expr1 [NOT] BETWEEN arithmetic-expr2 a arithmetic-expr3 operátor porovnání:
  - `Age BETWEEN 15 and 19` je ekvivalentní s `age>= 15 AND age <= 19`.
  - `Age NOT BETWEEN 15 and 19` je ekvivalentní s `age < 15 OR age > 19`.
  - Má-li některý z výrazů operace BETWEEN hodnotu NULL, je hodnota operace false. Je-li některý z výrazů operace `NOT BETWEEN` NULL, hodnota operace je true.
- Identifikátor `[NOT] IN (string-literal1, string-literal2,...)` operátoru porovnání, kde identifikátor má hodnotu typu String nebo NULL .
  - `Country IN ('UK', 'US', 'France')` je true pro `'UK'` a false pro `'Peru'`. Je ekvivalentní výrazu `(Country = 'UK') OR (Country = 'US') OR (Country = 'France')`.
  - `Country NOT IN ('UK', 'US', 'France')` je false pro `'UK'` a true pro `'Peru'`. Je ekvivalentní výrazu `NOT ((Country = 'UK') OR (Country = 'US') OR (Country = 'France'))`.
  - Je-li identifikátor operace IN nebo `NOT IN` NULL, hodnota operace je neznámá.
- `identifier [NOT] LIKE` *`pattern-value`* `[ESCAPE` *`escape-character`* `]` porovnávacího operátoru, kde `identifier` má řetězcovou hodnotu. *pattern-value* je řetězcový literál, kde _ zastupuje libovolný jednotlivý znak a % zastupuje libovolnou posloupnost znaků (včetně prázdné posloupnosti). Všechny ostatní znaky jsou stojan pro sebe. Volitelný řídicí znak *escape-character* je jednoznakový řetězcový literál, který se používá k úniku speciálního významu _ a % v *vzor-vzor*. Operátor LIKE musí být použit pouze k porovnání dvou řetězcových hodnot.
  - `phone LIKE '12%3'` je true pro 123 a 12993 a false pro 1234.
  - `word LIKE 'l_se'` je true pro ztrátu a false pro uvolnění.
  - `underscored LIKE '\_%' ESCAPE '\'` je true pro _foo a false pro bar.
  - `phone NOT LIKE '12%3'` je false pro 123 a 12993 a true pro 1234.
  - Je-li identifikátor operace LIKE nebo `NOT LIKE` NULL, hodnota operace je neznámá.

  Poznámka: Operátor LIKE musí být použit k porovnání dvou řetězcových hodnot. Hodnota parametru `Root.MQMD.CorrelId` je 24bajtové pole bajtů, nikoli znakový řetězec. Řetězec selektoru `Root.MQMD.CorrelId LIKE 'ABC%'` je akceptován syntaktickým analyzátorem jako syntakticky platný, ale je vyhodnocen na hodnotu false. Když porovnáváte bajtové pole se znakovým řetězcem, LIKE proto nemůže být použit.
- `identifier IS NULL` porovnávacího operátoru porovnání pro hodnotu pole záhlaví NULL nebo chybějící hodnotu vlastnosti.
- `identifier IS NOT NULL` porovnávacího operátoru testů pro existenci hodnoty pole záhlaví bez hodnoty null nebo hodnoty vlastnosti.
- Hodnoty null

  Vyhodnocení výrazů selektoru, které obsahují hodnoty NULL , je definováno sémantikou SQL 92 NULL , v souhrnu:
  - SQL považuje hodnotu NULL za neznámou.
  - Porovnání nebo aritmetika s neznámou hodnotou vždy poskytne neznámou hodnotu.

– Operátory `IS NULL` a `IS NOT NULL` převádějí neznámou hodnotu na hodnoty `TRUE` a `FALSE` .

Booleovské operátory používají tříhodnotovou logiku (`T=TRUE, F=FALSE, U=UNKNOWN`)

Tabulka 1. Výsledek logického operátoru, je-li logika `A AND B`		
Operátor A	**Operátor B**	**Výsledek (A AND B)**
T	F	F
T	U	U
T	T	T
F	T	F
F	U	F
F	F	F
U	T	U
U	U	U
U	F	F

Tabulka 2. Výsledek logického operátoru, je-li logika `A OR B`		
Operátor A	**Operátor B**	**Výsledek (`A OR B`)**
T	F	T
T	U	T
T	T	T
F	T	T
F	U	U
F	F	F
U	T	T
U	U	U
U	F	U

Tabulka 3. Výsledek logického operátoru, je-li logika `NOT A`	
Operátor A	**Výsledek (`NOT A`)**
T	F
F	T
U	U

Následující selektor zpráv vybírá zprávy s typem zprávy auto, barvou modré barvy a váhou větší než 2500 lbs:

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

Ačkoli SQL podporuje fixní desetinné porovnání a aritmetiku, selektory zpráv nikoli. To je důvod, proč jsou přesné číselné literály omezené na ty, které nemají desetinné číslo. Je také důvod, proč existují číselné hodnoty s desetinnou hodnotou jako alternativní znázornění přibližné numerické hodnoty.

Komentáře SQL nejsou podporovány.

Související pojmy

“Vlastnosti zprávy” na stránce 21

Pomocí vlastností zprávy můžete aplikaci umožnit vybrat zprávy ke zpracování nebo načíst informace o zprávě bez přístupu k záhlavím MQMD nebo MQRFH2 . Usnadňují také komunikaci mezi aplikacemi IBM MQ a JMS .

Související informace

MsgHandle

MQBUFMH-Převedení vyrovnávací paměti na popisovač zprávy

Pravidla řetězce výběru a omezení

Seznamte se s těmito pravidly o tom, jak jsou řetězce výběru interpretovány a znaková omezení, abyste se vyhnuli možným problémům při použití selektorů.

- Výběr zpráv pro systém zpráv publikování/odběru se provádí ve zprávě, jak ji odesílá vydavatel. Viz Výběrové řetězce.
- Rovnocennost je testována pomocí jednoho znaku rovnítka; například `a = b` je správný, zatímco `a == b` je chybný.
- Operátor, který používá mnoho programovacích jazyků k reprezentaci 'not equal to', je `!=`. Tato reprezentace není platným synonymem pro <> ; Například `a <> b` je platný, zatímco `a != b` je neplatný.
- Jednoduché uvozovky jsou rozpoznávány pouze v případě, že ' (U+0027) je použit znak. Podobně dvojité uvozovky jsou platné pouze v případě, že jsou použity k uzavření bajtových řetězců, musí být použity " (U+0022).
- Symboly &, &&, | a || nejsou synonymy pro logickou konjunkci a disjunkci; například `a && b` musí být zadáno jako `a AND b`.
- Zástupné znaky * a ? nejsou synonymy pro % a _.
- Selektory obsahující složené výrazy jako `20 < b < 30` nejsou platné. Syntaktický analyzátor vyhodnocuje operátory, které mají stejnou prioritu než zleva doprava. Tento příklad by se tedy stal `(20 < b) < 30`, což nedává smysl. Namísto toho musí být výraz napsán jako `(b > 20) AND (b < 30)`.
- Řetězce v bajtech musí být uzavřeny v uvozovkách; pokud jsou použity jednoduché uvozovky, bude bajtový řetězec brát jako řetězcový literál. Počet znaků (nikoli počet znaků, které znaky představují) za znakem `0x` musí být násobkem dvou.
- Klíčové slovo `IS` není synonymem pro znak rovnítka. Výběrové řetězce a `IS 3` a `b IS 'red'` tedy nejsou platné. Klíčové slovo `IS` existuje pouze pro podporu případů `IS NULL` a `IS NOT NULL` .

Související pojmy

“Chování výběru” na stránce 29

Přehled chování výběru produktu IBM MQ .

Související informace

Výběrové řetězce

Aspekty UTF-8 a Unicode při použití selektorů zpráv

Znaky, které nejsou uzavřeny v jednoduchých uvozovkách, které tvoří vyhrazená klíčová slova řetězce výběru, musí být zadány v jazyku Basic Latin Unicode (řazeno od znaku U+0000 až U+0007F). Není platný pro použití jiných znázornění alfanumerických znaků v jiných kódových bodech. Například číslo 1 musí být vyjádřeno jako U+0031 v Unicode, není platné pro použití ekvivalentního znaku plné šířky U+FF11 nebo arabského ekvivalentu U+0661.

Názvy vlastností zprávy mohou být zadány pomocí libovolné platné posloupnosti znaků Unicode. Názvy vlastností zprávy obsažené ve výběrových řetězcích, které jsou kódovány v UTF-8 , budou ověřeny i v případě, že obsahují vícebajtové znaky. Ověření vícebajtové znakové sady UTF-8 je striktní a musíte zajistit, aby pro názvy vlastností zpráv byly použity platné posloupnosti UTF-8 . V 9.0.0 Znaky nad rámec Unicode Basic Multilingual Plane (ty, které jsou nad U + FFFF), představované v souboru UTF-16

pomocí náhradních kódových bodů (X'D800'až X'DFFF'), nebo 4 bajty v UTF-8, nejsou v názvech vlastností zpráv podporovány.

Při porovnávání s rovností se neprovádí žádné další zpracování na názvech vlastností nebo hodnotách. To znamená například, že žádné pre/de-composition se koná a ligattures nejsou uvedeny žádné zvláštní význam. Například, předkomponovaný znak přehlásku U+00FC není považován za ekvivalent U+0075 + U+0308 a znaková sekvence se nepovažuje za ekvivalent Unicode U+FB00 (LATIN SME LIGATURE FF).

Data vlastností uzavřená v jednoduchých uvozovkách mohou být reprezentována libovolným sekvencí bajtů a nejsou ověřována.

Výběr obsahu zprávy

Je možné se přihlásit k odběru obsahu informačního obsahu zprávy (označovaného také jako filtrování obsahu), ale rozhodnutí o tom, které zprávy mají být doručeny takovému odběru, nemůže být provedeno přímo produktem IBM MQ; namísto rozšířeného poskytovatele výběru zpráv, například IBM Integration Bus, je pro zpracování zpráv nezbytný.

Když aplikace publikuje na řetězec tématu, kde jeden nebo více odběratelů má výběrový řetězec výběru obsahu zprávy, produkt IBM MQ požádá poskytovatele rozšířeného výběru zpráv o syntaktické analýze publikace a informuje IBM MQ o tom, zda publikování odpovídá kritériím výběru určeným každým odběratelem s filtrem obsahu.

Pokud poskytovatel výběru rozšířených zpráv zjistí, že se publikování shoduje s výběrovým řetězcem odběratele, zpráva bude i nadále doručována odběrateli.

Pokud poskytovatel výběru rozšířených zpráv zjistí, že se publikování neshoduje, zpráva se odběrateli nebude doručovat. To může vést k selhání volání MQPUT nebo MQPUT1 s kódem příčiny MQRC_PUBLICATION_FAILURE. Pokud poskytovatel rozšířeného výběru zpráv nemůže provést analýzu publikování, je vrácen kód příčiny MQRC_CONTENT_ERROR a volání MQPUT nebo MQPUT1 selže.

Pokud poskytovatel rozšířených výběru zpráv není k dispozici nebo není schopen určit, zda by měl odběratel obdržet publikování, je vrácen kód příčiny MQRC_SELECTION_NOT_AVAILABLE a volání MQPUT nebo MQPUT1 selže.

Je-li vytvořen odběr s filtrem obsahu a poskytovatel rozšířeného výběru zpráv není k dispozici, volání MQSUB selže s kódem příčiny MQRC_SELECTION_NOT_AVAILABLE. Je-li obnoven odběr s filtrem obsahu a poskytovatel výběru rozšířených zpráv není k dispozici, volání MQSUB vrátí varování MQRC_SELECTION_NOT_AVAILABLE, ale odběr je povolen pro pokračování.

Související informace

Výběrové řetězce

Asynchronní spotřeba zpráv produktu IBM MQ

Asynchronní spotřeba používá sadu rozšíření rozhraní MQI (Message Queue Interface), volání MQI MQCB a MQCTL, která umožňují zápis aplikace MQI k přijímání zpráv ze sady front. Zprávy jsou doručovány do aplikace vyvoláním 'jednotky kódu', identifikované aplikací předáním zprávy nebo tokenu představujícím zprávu.

V nejpřímějším prostředí aplikace je jednotka kódu definována ukazatelem funkce, avšak v jiných prostředích může být jednotka kódu definována pomocí názvu programu nebo modulu.

V asynchronní spotřebě zpráv se používají následující termíny:

Spotřebitel zpráv.
Konstrukt programování, který umožňuje definovat program nebo funkci, který má být vyvolán se zprávou, když je k dispozici jeden vyhovující požadavek na aplikace.

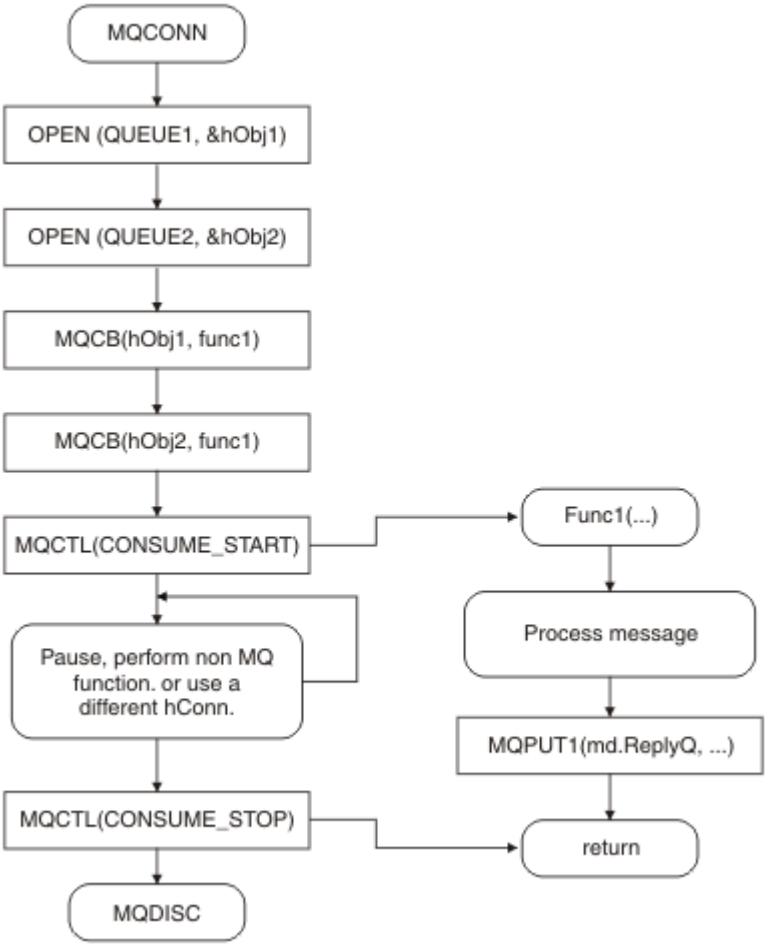
obslužná rutina událostí
Konstrukt programování umožňující definovat program nebo funkci, která má být vyvolána při výskytu asynchronní události, jako je například uvedení správce front do klidového stavu, nebo funkce.

Zpětné volání
Generický termín používaný k odkazování na rutinu Message Consumer nebo obslužné rutiny událostí.

Asynchronní spotřeba může zjednodušit návrh a implementaci nových aplikací, zejména těch, které zpracovávají více vstupních front nebo odběrů. Pokud však používáte více než jednu vstupní frontu a zpracovával zprávy v posloupnosti priorit, je posloupnost priorit sledována nezávisle v každé frontě: můžete dostat zprávy s nízkou prioritou z jedné fronty před zprávami s vysokou prioritou od druhé. Pořadí zpráv v rámci více front není garantováno. Všimněte si také, že pokud používáte uživatelské procedury rozhraní API, může být zapotřebí, abyste je změnili, aby zahrnovaly volání MQCB a MQCTL.

Následující ilustrace poskytují příklad, jak můžete tuto funkci použít.

Produkt Obrázek 5 na stránce 37 zobrazuje zprávy s vícenásobnými aplikacemi spotřebovávající zprávy ze dvou front. Příklad zobrazuje všechny zprávy doručené do jedné funkce.

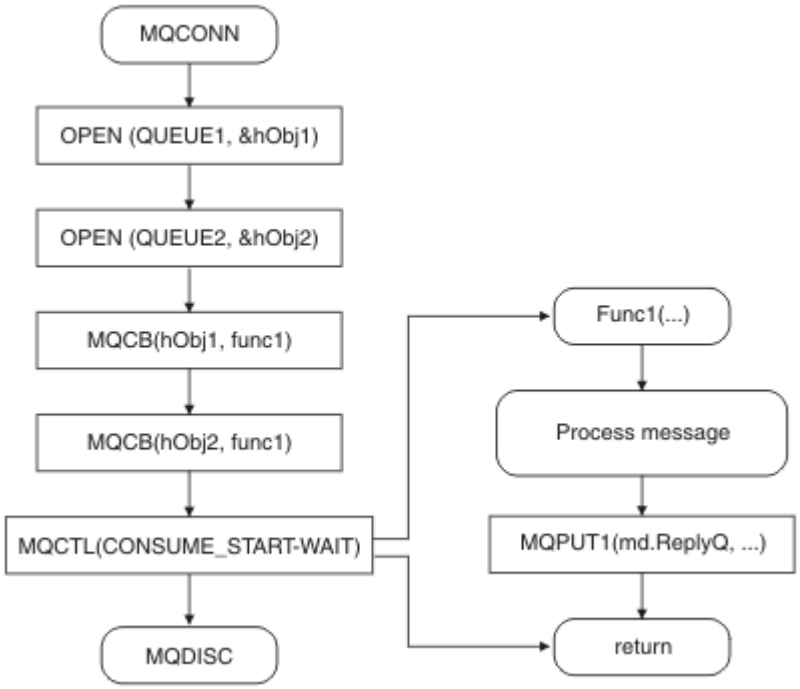


Obrázek 5. Standardní aplikace řízená zprávami spotřebovávající ze dvou front

z/OS V systému z/OSmusí hlavní řídicí podproces vydat volání MQDISC před ukončením. Tímto způsobem lze všechny podprocesy zpětného volání ukončit a uvolnit systémové prostředky.

Obrázek 6 na stránce 38 Tento ukázkový tok zobrazuje jednovláknovou aplikaci spotřebovávající zprávy ze dvou front. Příklad zobrazuje všechny zprávy doručené do jedné funkce.

Rozdíl mezi asynchronním případem je ten, že se řízení nevrátí k emitentovi MQCTL, dokud se všichni spotřebitelé deaktivují. To znamená, že jeden odběratel vydal požadavek MQCTL STOP nebo uvedení správce front do klidového stavu.



Obrázek 6. Jednotlivý Threaded Message Driven application consuming from two queues

Skupiny zpráv

Zprávy se mohou vyskytnout uvnitř skupin, aby bylo možné pořadí zpráv.

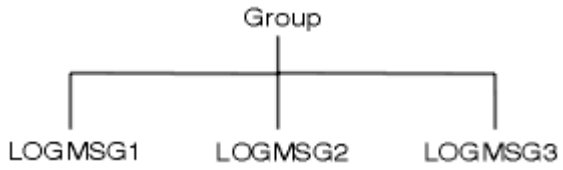
Skupiny zpráv umožňují označení více zpráv jako vzájemně souvisejících a logických příkazů, které mají být použity ve skupině (viz "Logické a fyzické uspořádání" na stránce 743). Na Multiplatformy, segmentace zpráv umožňuje rozdělení velkých zpráv do menších segmentů. Seskupené nebo segmentované zprávy nelze použít při vkládání do tématu.

Hierarchie v rámci skupiny je následující:

Skupina
Jedná se o nejvyšší úroveň v hierarchii a je identifikována pomocí *GroupId*. Skládá se z jedné nebo více zpráv, které obsahují stejné *GroupId*. Tyto zprávy mohou být uloženy kdekoli ve frontě.

Poznámka: Termín *zpráva* se zde používá k označení jedné položky ve frontě, jako by byla vrácena jediná MQGET, která neurčuje MQGMO_COMPLETE_MSG.

Obrázek 7 na stránce 38 zobrazuje skupinu logických zpráv:



Obrázek 7. Skupina logických zpráv

Když otevřete frontu a uvedete MQOO_BIND_ON_GROUP, vynutí odeslání všech zpráv ve skupině, které jsou odeslány do této fronty, do stejné instance fronty. Další informace o volbě BIND_ON_GROUP naleznete v části Ošetřování afinit zprávy.

Logická zpráva
Logické zprávy ve skupině jsou identifikovány v polích *GroupId* a *MsgSeqNumber* . Hodnota *MsgSeqNumber* začíná v 1 pro první zprávu v rámci skupiny a pokud zpráva není ve skupině, hodnota pole je 1.

Použít logické zprávy v rámci skupiny k:

- Zajistěte objednávání (pokud to není zaručeno za okolností, za kterých se zpráva přenáší).
- Povolit aplikacím seskupovat podobné zprávy (například ty, které musí být všechny zpracovány stejnou instancí serveru).

Každá zpráva v rámci skupiny se skládá z jedné fyzické zprávy, pokud není rozdělena na segmenty. Každá zpráva je logicky samostatná zpráva a pouze pole *GroupId* a *MsgSeqNumber* v deskriptoru MQMD musí nést jakýkoli vztah k ostatním zprávám ve skupině. Ostatní pole v produktu MQMD jsou nezávislá; některá by mohla být identická pro všechny zprávy ve skupině, zatímco jiné mohou být odlišné. Zprávy ve skupině mohou mít například různé názvy formátů, CCSID a kódování.

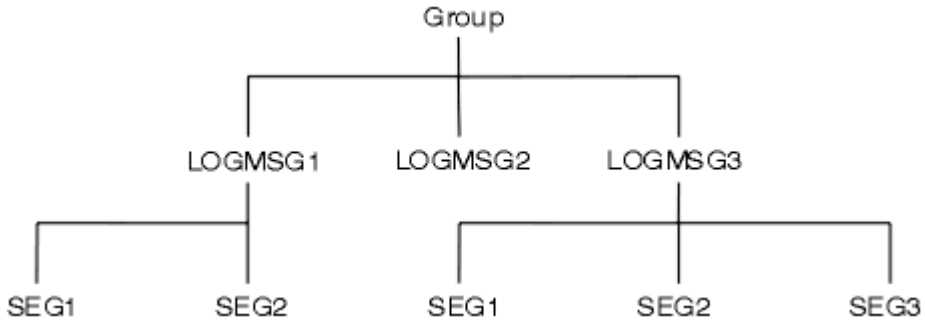
Segment

Segmenty se používají ke zpracování zpráv, které jsou příliš velké pro vložení nebo získání aplikace nebo správce front (včetně zasahujících správců front, kterými se zpráva předává). Další informace viz “Segmentace zpráv” na stránce 759.

Jednotlivá zpráva je rozdělena do menších zpráv nazývaných *segmenty*. Segment zprávy je identifikován pomocí polí *GroupId*, *MsgSeqNumber*a *Offset* . Pole *Offset* začíná na nule pro první segment ve zprávě.

Každý segment se skládá z jedné fyzické zprávy, která může patřit do skupiny (Obrázek 8 na stránce 39 ukazuje příklad zpráv uvnitř skupiny). Segment je logicky součástí jediné zprávy, takže pouze pole *MsgId*, *Offset*a *MsgFlags* v deskriptoru MQMD by se měly lišit mezi oddělenými segmenty stejné zprávy. Pokud se nezdaří doručení segmentu, vrátí se podle potřeby kód příčiny MQRC_INCOMPLEE_GROUP nebo MQRC_INCOMPLEE_MSG .

Obrázek 8 na stránce 39 zobrazuje skupinu logických zpráv, z nichž některé jsou segmentovány:



Obrázek 8. Segmentované zprávy

z/OS Segmentace není v produktu IBM MQ for z/OSpodporována.

Segmentované nebo seskupené zprávy nemůžete používat s publikováním/odběrem.

Související pojmy

“Segmentace zpráv” na stránce 759
Tyto informace použijte k seznámení se s segmentováním zpráv. Tato funkce není v produktu IBM MQ for z/OS nebo v aplikacích využívajících produkt IBM MQ classes for JMSpodporována.

Související odkazy

“Logické a fyzické uspořádání” na stránce 743
Zprávy ve frontách se mohou vyskytnout (v rámci každé úrovně priority) ve *fyzickém* nebo *logickém* pořadí.

Související informace

MQMD-deskriptor zprávy

Trvalost zpráv

Trvalé zprávy se zapisují do protokolů a datových souborů fronty. Je-li správce front restartován po selhání, obnoví tyto trvalé zprávy podle potřeby z protokolovaných dat. Zprávy, které nejsou trvalé, jsou zahozeny v případě zastavení správce front bez ohledu na to, zda je příkaz stoppage uveden jako výsledek příkazu operátora nebo kvůli selhání některé části systému.

z/OS Netrvalé zprávy uložené ve spojovacím zařízení (CF) v systému z/OS jsou výjimkou této události. Přetrvávají tak dlouho, dokud prostředek CF zůstane dostupný.

Pokud při vytváření zprávy inicializujete deskriptor zprávy (MQMD) s použitím výchozích hodnot, bude perzistence pro zprávu převzata z atributu **DefPersistence** fronty určené v příkazu MQOPEN. Eventuálně můžete nastavit trvání zprávy pomocí pole *Persistence* struktury MQMD pro definování zprávy jako trvalé nebo přechodné.

Výkon aplikace je ovlivněn, když používáte trvalé zprávy; rozsah efektu závisí na charakteristice výkonu subsystému I/O počítače a na tom, jak budete používat volby synchronizačního bodu na každé platformě:

- Trvalá zpráva, mimo aktuální jednotku práce, se zapisuje na disk při každé operaci put a get. Viz "Potvrzení a zálohování jednotek práce" na stránce 818.
- ULW z/OS Pro všechny platformy kromě produktu IBM ise trvalá zpráva v rámci aktuální jednotky práce protokoluje pouze tehdy, je-li jednotka práce potvrzena a jednotka práce může obsahovat mnoho operací fronty.

Přechodné zprávy lze použít pro rychlé zasílání zpráv. Další informace o rychlých zprávách viz Bezpečnost zpráv .

Poznámka: Kombinace zápisu trvalých zpráv v rámci pracovní jednotky a zápisu trvalých zpráv mimo jednotku nebo do práce může způsobit potenciálně závažné problémy s výkonem vašich aplikací. To platí zejména tehdy, je-li pro obě operace použita stejná cílová fronta.

Zprávy, které se nepodařilo doručit

Když správce front nemůže vložit zprávu do fronty, máte různé možnosti.

Můžete provést následující akce:

- Pokuste se znovu vložit zprávu do fronty.
- Vyžádejte si zprávu, že zpráva byla vrácena odesílateli.
- Vložte zprávu do fronty nedoručených zpráv.

Další informace naleznete v dokumentu"Obsluha chyb procedurálních programů" na stránce 1015.

Zprávy, které jsou zálohovány

Při zpracování zpráv z fronty pod kontrolou jednotky práce se může jednotka práce skládat z jedné nebo více zpráv. Dojde-li k odvolání, budou zprávy, které byly načteny z fronty, znovu zavedeny do fronty a mohou být zpracovány znovu v jiné pracovní jednotce. Pokud je zpracování určité zprávy příčinou problému, je jednotka práce znovu vrácena. To může způsobit smyčku zpracování. Zprávy, které byly vloženy do fronty, byly odebrány z fronty.

Aplikace dokáže detekovat zprávy, které jsou zachyceny v této smyčce testováním pole *BackoutCount* v produktu MQMD. Aplikace může buď opravit situaci, nebo vydat varování operátorovi.

Multi Počet odvolání vždy přežije restarty správce front. Jakákoli změna atributu **HardenGetBackout** se ignoruje.

z/OS U sdílených front počet vrácení vždy přežije restarty správce front. Pro všechny ostatní konfigurace v systému z/OSse ujistěte, že počet vrácení pro soukromé fronty přežije restarty správce front, nastavte atribut *HardenGetBackout* na hodnotu MQQA_BACOUT_HARDENED; v opačném případě, pokud se má správce front restartovat, nezachová se pro každou zprávu přesný počet vrácení. Nastavení atributu tímto způsobem přidá náklady na další zpracování.

Další informace o potvrzování a vracení zpráv naleznete v tématu "Potvrzení a zálohování jednotek práce" na stránce 818.

Fronta pro zařazení do fronty a správce front

Vyskytly se případy, kdy můžete obdržet zprávy jako odpověď na odeslalou zprávu:

- Odpověď na zprávu požadavku v odpovědi na zprávu požadavku
- Zpráva sestavy o neočekávané události nebo vypršení platnosti
- Zpráva se sestavou o události COA (Potvrzení příjmu) nebo COD (Potvrzení doručení)
- Zpráva se sestavou týkající se PAN (Pozitivní akce na akci) nebo události NAN (Negative Action Notification)

Pomocí struktury MQMD zadejte do pole *ReplyToQ* název fronty, do které chcete odesílat zprávy reply a zprávy. Do pole *ReplyToQMgr* zadejte název správce front, který vlastní frontu pro odpovědi.

Ponecháte-li pole *ReplyToQMgr* prázdné, nastaví správce front obsah následujících polí ve frontě zpráv ve frontě:

ReplyToQ
Je-li *ReplyToQ* lokální definicí vzdálené fronty, pole *ReplyToQ* je nastaveno na název vzdálené fronty; jinak se toto pole nezmění.

ReplyToQMgr
Je-li *ReplyToQ* lokální definicí vzdálené fronty, je pole *ReplyToQMgr* nastaveno na název správce front, který vlastní vzdálenou frontu; v opačném případě je pole *ReplyToQMgr* nastaveno na název správce front, ke kterému je aplikace připojena.

Poznámka: Můžete požádat správce front o více pokusů o doručení zprávy a můžete požádat, aby byla zpráva vyřazena, pokud selže. Není-li zpráva po selhání k doručení vyřazena, vzdálený správce front vloží zprávu do fronty nedoručených zpráv (viz "Použití fronty nedoručených zpráv (nedoručená zpráva)" na stránce 1019).

kontext zprávy

Informace *Kontext zprávy* umožňují aplikaci, která načte zprávu, zjistit informace o odesílateli zprávy.

Načítání aplikace může vyžadovat:

- Zkontrolujte, zda má odesílající aplikace správnou úroveň oprávnění.
- Provedení určité funkce účtování, aby mohla účtovat odesílající aplikaci za jakoukoli práci, kterou musí provést
- Uchovat záznam pro audit všech zpráv, se kterými pracoval.

Použijete-li volání MQPUT nebo MQPUT1 k vložení zprávy do fronty, můžete určit, že správce front má přidat některé výchozí kontextové informace do deskriptoru zpráv. Aplikace, které mají odpovídající úroveň oprávnění, mohou přidávat další informace o kontextu. Další informace o tom, jak zadat informace o kontextu, viz "Řízení informací o kontextu zprávy" na stránce 729.

Kontext uživatele je používán správcem front při generování následujících typů zpráv sestavy:

- Potvrdit při doručení
- Vypršení

Když jsou tyto zprávy sestavy generovány, je kontext uživatele kontrolován pro + put a + passid authority na místo určení sestavy. Pokud má kontext uživatele nedostatečné oprávnění, umístí se zpráva sestavy do fronty nedoručených zpráv, je-li definována. Není-li fronta nedoručených zpráv, bude zpráva sestavy vyřazena.

Všechny kontextové informace jsou uloženy v kontextových polích deskriptoru zpráv. Typ informací spadá do kontextu identity, původu a kontextu uživatele.

kontext identity

Informace *Kontext identity* identifikují uživatele aplikace, který poprvé vložil zprávu do fronty. Suitatelně autorizované aplikace mohou nastavit následující pole:

- Správce front vyplní pole *UserIdentifier* názvem, který identifikuje uživatele. Způsob, jakým správce front může toto provést, závisí na prostředí, ve kterém je aplikace spuštěna.
- Správce front vyplní pole *AccountingToken* token nebo číslo, které určil z aplikace, která vložila zprávu.
- Aplikace mohou použít pole *ApplIdentityData* pro jakékoli další informace, které chtějí zahrnout o uživateli (například zašifrované heslo).

Identifikátor zabezpečení systému Windows (SID) je uložen v poli *AccountingToken* , je-li vytvořena zpráva pod IBM MQ for Windows. Identifikátor SID lze použít k doplnění pole *UserIdentifier* a k zavedení pověření uživatele.

Informace o tom, jak správce front vyplní pole *UserIdentifier* a *AccountingToken* , naleznete v popisech těchto polí v části UserIdentifier a AccountingToken.

Aplikace, které předávají zprávy z jednoho správce front do jiného, by měly také předávat informace o kontextu identity, aby jiné aplikace znali identitu původce zprávy.

Původní kontext

Informace *Kontext původního stavu* popisují aplikaci, která vložila zprávu do fronty, kde je zpráva momentálně uložena. Deskriptor zprávy obsahuje následující pole pro informace o kontextu původu:

- *PutApplType* definuje typ aplikace, která vložila tuto zprávu (například transakce CICS).
- *PutApplName* definuje název aplikace, která vložila tuto zprávu (například jméno úlohy nebo transakce).
- *PutDate* definuje datum, kdy byla zpráva vložena do fronty.
- *PutTime* definuje čas, ve kterém byla zpráva vložena do fronty.
- *ApplOriginData* definuje jakékoli další informace, které chce aplikace zahrnout o původu zprávy. Může být například nastaven vhodně autorizovanými aplikacemi, které označují, zda jsou data identity důvěryhodná.

Informace o kontextu původu jsou obvykle zadávané správcem front. Greenwichský střední čas (GMT) se používá pro pole *PutDate* a *PutTime* . Viz popisy těchto polí v polích PutDate a PutTime.

Aplikace s dostatečným oprávněním může poskytovat svůj vlastní kontext. To umožňuje zachování účetních informací, když má jeden uživatel odlišné ID uživatele na každém systému, který zpracovává zprávu, ze které pocházejí.

Objekty IBM MQ

Tyto informace poskytují podrobné informace o objektech produktu IBM MQ , které zahrnují: správce front, skupiny sdílení front, fronty, objekty administrativních témat, seznamy názvů, definice procesů, objekty ověřovacích informací, kanály, paměťové třídy, moduly listener a služby.

Správci front definují vlastnosti těchto objektů (známé jako atributy). Hodnoty těchto atributů ovlivňují způsob, jakým produkt IBM MQ zpracovává tyto objekty. Ve svých aplikacích můžete tyto objekty řídit pomocí rozhraní MQI (Message Queue Interface). Objekty jsou identifikovány *deskriptorem objektu* (MQOD), jsou-li adresovány z programu.

Použijete-li příkazy IBM MQ k definování, úpravě nebo odstranění objektů, například správce front zkontroluje, zda máte k provedení těchto operací požadovanou úroveň oprávnění. Podobně platí, že pokud aplikace používá volání MQOPEN k otevření objektu, správce front zkontroluje, zda má aplikace požadovanou úroveň oprávnění před tím, než povolí přístup k danému objektu. Kontroly jsou prováděny na jménu otevíraný objekt.

Související pojmy

“Řízení informací o kontextu zprávy” na stránce 729
Použijete-li volání MQPUT nebo MQPUT1 k vložení zprávy do fronty, můžete určit, že správce front má přidat některé výchozí kontextové informace do deskriptoru zpráv. Aplikace, které mají odpovídající

úroveň oprávnění, mohou přidávat další informace o kontextu. Pole voleb ve struktuře MQPMO můžete použít k řízení informací o kontextu.

Související odkazy

“Volby MQOPEN vztahující se ke kontextu zprávy” na stránce 720
Chcete-li mít možnost přidružit kontextové informace ke zprávě, když ji vložíte do fronty, musíte při otevření fronty použít jednu z voleb kontextu zprávy.

Příprava a spuštění aplikací serveru Microsoft Transaction Server

Chcete-li připravit aplikaci MTS ke spuštění jako aplikaci produktu IBM MQ MQI client , postupujte podle těchto pokynů, jak je to vhodné pro vaše prostředí.

Obecné informace o tom, jak vyvíjet aplikace MTS (Microsoft Transaction Server), které přistupují k prostředkům produktu IBM MQ , najdete v části týkající se MTS v Centru nápovědy IBM MQ .

Chcete-li připravit aplikaci MTS ke spuštění jako aplikaci produktu IBM MQ MQI client , proveďte jednu z následujících komponent pro každou komponentu aplikace:

- Pokud komponenta používá vazby jazyka C pro rozhraní MQI, postupujte podle pokynů v části “Příprava programů jazyka C v produktu Windows” na stránce 994 , ale propojte ji s knihovnou mqicxa.lib místo mqic.lib.
- Pokud komponenta používá třídy C++ IBM MQ , postupujte podle pokynů v “Sestavování programů C++ v systému Windows” na stránce 510 , ale propojte komponentu s knihovnou imqx23vn.lib místo imqc23vn.lib.
- Pokud komponenta používá vazby jazyka Visual Basic pro rozhraní MQI, postupujte podle pokynů v příručce “Příprava programů produktu Visual Basic v produktu Windows” na stránce 998 , ale při definování projektu Visual Basic zadejte do pole **Argumenty podmíněné kompilace** hodnotu `MqType=3` .
- Pokud komponenta používá třídy automatizace IBM MQ pro ActiveX (MQAX), definujte proměnnou prostředí GMQ_MQ_LIB s hodnotou `mqic32xa.dll`.

  Proměnnou prostředí můžete definovat ve své aplikaci, nebo ji můžete definovat tak, aby její rozsah byl celý systém. Definováním v celém systému však může dojít k nesprávnému chování existující aplikace MQAX, která nedefinuje proměnnou prostředí v rámci aplikace.

Použití IBM MQ s WebSphere Application Server

Toto téma vám pomůže porozumět použití produktu IBM MQ s produktem WebSphere Application Server.

Aplikace, které jsou zapsány v produktu Java a jsou spuštěny pod WebSphere Application Server , mohou používat specifikaci Java Messaging Service (JMS) k provádění systému zpráv. Systém zpráv v tomto prostředí může být poskytován správcem front IBM MQ .

Výhodou použití správce front produktu IBM MQ je to, že připojení aplikací produktu JMS se může plně podílet na funkčnosti sítě IBM MQ , což umožňuje aplikacím vyměňovat si zprávy se správci front spuštěnými na velkém počtu platforem.

Aplikace mohou pro objekt továrny připojení fronty použít buď *přenos klienta* , nebo *přenos vazeb* . Pro *přenos vazeb*musí správce front existovat lokálně na aplikaci, která vyžaduje připojení.

Zprávy produktu JMS ve frontách produktu IBM MQ standardně používají záhlaví MQRFH2 k ukládání některých informací záhlaví zprávy produktu JMS . Mnoho starších aplikací produktu IBM MQ nemůže zpracovávat zprávy s těmito záhlavími a vyžadovat jejich vlastní záhlaví charakteristik, například MQCIH pro most CICS Bridge nebo MQWIH pro aplikace IBM MQ Workflow. Další informace o těchto speciálních aspektech naleznete v tématu Mapování zpráv produktu JMS na zprávy produktu IBM MQ.

Aspekty návrhu pro aplikace produktu IBM MQ

Když se rozhodnete, jak vaše aplikace mohou využívat výhody platforem a prostředí, které máte k dispozici, musíte se rozhodnout, jak používat funkce nabízené produktem IBM MQ.

Při návrhu aplikace produktu IBM MQ vezměte v úvahu následující otázky a volby:

Typ aplikace
Jaký je účel vaší aplikace? Informace o různých typech aplikací, které můžete vyvinout, najdete v následujících odkazech:

- Server
- Klient
- Publikování/odběr
- Webové služby.
- Uživatelské procedury, uživatelské procedury rozhraní API a instalovatelné služby

Kromě toho můžete také napsat své vlastní aplikace k automatizaci administrace produktu IBM MQ. Další informace naleznete v tématu Rozhraní MQAI (IBM MQ Administration Interface) a v tématu Automatizace administrativních úloh.

Programovací jazyk
IBM MQ podporuje řadu procedurálních a objektově orientovaných programovacích jazyků pro psaní aplikací. Další informace viz “Vyvíjení aplikací pro IBM MQ” na stránce 5.

Aplikace pro více platforem

Bude vaše aplikace spuštěna na více než jedné platformě? Máte strategii pro přechod na jinou platformu od té, kterou používáte dnes? Je-li odpověď na jednu z těchto otázek ano, ujistěte se, že jste naprogramovat své programy pro nezávislost na platformě.

Používáte-li např. C, kód ANSI standard C. Používejte standardní funkce knihovny jazyka C a nikoli ekvivalentní funkce specifické pro platformu, a to i v případě, že funkce specifická pro danou platformu je rychlejší nebo efektivnější. Výjimkou je, když je účinnost v kódu prvořadá, pokud byste měli zadat kód pro obě situace pomocí #ifdef. Příklad:

```
#ifdef _AIX
    AIX specific code
#else
    generic code
#endif
```

Typy front
Chcete vytvořit frontu pokaždé, když ji budete potřebovat, nebo chcete použít fronty, které již byly nastaveny? Chcete odstranit frontu, když jste ji již dokončili, nebo chcete frontu použít znovu? Chcete použít alias fronty pro nezávislost aplikace? Chcete-li zjistit, které typy front jsou podporovány, prostudujte si téma Fronty.

z/OS **Použití sdílených front, skupin sdílení front a klastrů skupin sdílení front (pouze IBM MQ for z/OS)**
Možná budete chtít využívat výhod zvýšené dostupnosti, rozšiřitelnosti a vyvážení pracovní zátěže, které jsou možné při použití sdílených front se skupinami sdílení front. Další informace naleznete v tématu Sdílené fronty a skupiny sdílení front .

Můžete také chtít odhadnout průměrné a špičkové toky zpráv a zvážit použití klastrů skupin sdílení front k rozložení pracovní zátěže. Další informace naleznete v tématu Sdílené fronty a skupiny sdílení front .

Použití klastrů správců front
Možná budete chtít využít zjednodušené administrace systému a zvýšit dostupnost, rozšiřitelnost a vyrovnávání pracovní zátěže, které jsou možné, když používáte klastry.

Typy zpráv
Možná budete chtít použít datagramy pro jednoduché zprávy, ale požadovat zprávy (pro které očekáváte odpovědi) pro jiné situace. Možná budete chtít některé z vašich zpráv přiřadit jiné priority. Další informace o návrhu zpráv viz “Metody návrhu pro zprávy” na stránce 47.

Použití publikování/odběru nebo zasílání zpráv mezi dvěma body

Při použití systému zpráv publikování/odběru odesílá odesílající aplikace informace, které chce sdílet ve zprávě produktu IBM MQ do standardního cíle spravovaného produktem IBM MQ publish? subscribe, a umožňuje produktu IBM MQ zpracovat distribuci těchto informací. Cílová aplikace nemusí vědět nic o zdroji informací, které přijímá, ale registruje zájem o jedno nebo více témat a přijímá tyto informace, jakmile je k dispozici. Další informace o zasílání zpráv publikování/odběru naleznete v tématu Systém zpráv publikování/odběru.

Při použití systému zpráv mezi dvěma body odešle odesílající aplikace zprávu do určité fronty, odkud ví, že přijímající aplikace ji načte. Přijímající aplikace získává zprávy z určité fronty a pracuje s jejich obsahem. Aplikace bude často fungovat jako odesilatel i příjemce, odešle dotaz na jinou aplikaci a obdrží odpověď.

Řízení vašich programů IBM MQ

Možná budete chtít spustit některé programy automaticky nebo aby programy čekaly, dokud určitá zpráva nepřijde do fronty (pomocí funkce IBM MQ *triggering* , viz “Spuštění aplikací produktu IBM MQ pomocí spouštěčů” na stránce 829). Eventuálně můžete chtít spustit další instanci aplikace, když se zprávy ve frontě nezpracovávají dostatečně rychle (pomocí funkce IBM MQ *instrumentation events* , jak je popsáno v části Události instrumentace).

Spuštění vaší aplikace na klientovi IBM MQ

Úplná MQI je podporována v prostředí klienta a téměř každá aplikace IBM MQ zapsaná v procedurálním jazyce může být znovu předána ke spuštění na serveru IBM MQ MQI client. Propojte aplikaci na serveru IBM MQ MQI client s knihovnou MQIC a nikoli s knihovnou MQI. z/OS Získání (signál) na z/OS není podporováno.

Poznámka: Aplikace běžící na klientovi produktu IBM MQ se může připojit k více než jednomu správci front souběžně, nebo může použít název správce front s hvězdičkou (*) ve volání MQCONN nebo MQCONNX. Změňte aplikaci, pokud se chcete připojit ke knihovnám správce front místo knihoven klienta, protože tato funkce nebude k dispozici.

Další informace viz “Spuštění aplikací v prostředí produktu IBM MQ MQI client” na stránce 883.

Výkon aplikace

Konstrukční rozhodnutí mohou mít dopad na výkon aplikací, a to pro návrhy na zvýšení výkonu aplikací produktu IBM MQ , viz “Aspekty návrhu a výkonu aplikací” na stránce 48 IBM i a “Aspekty návrhu a výkonu pro aplikace produktu IBM i” na stránce 52 .

Rozšířené metody IBM MQ

Pro pokročilejší aplikace může být vhodné použít některé rozšířené metody IBM MQ , jako např. korelace odpovědí a generování a odesílání informací o kontextu produktu IBM MQ . Další informace viz “Metody návrhu pro rozšířené aplikace” na stránce 50.

Zabezpečení vašich dat a správa jeho integrity

Můžete použít informace o kontextu předané spolu se zprávou a otestovat, zda byla zpráva odeslána z přípustného zdroje. Zařízení syncpointing, kterou poskytuje produkt IBM MQ nebo váš operační systém, můžete použít k zajištění konzistence dat s ostatními prostředky (další podrobnosti viz “Potvrzení a zálohování jednotek práce” na stránce 818). Funkci *perzistence* zpráv produktu IBM MQ můžete použít k zajištění doručení důležitých zpráv.

Testování aplikací produktu IBM MQ

Vývojové prostředí aplikací pro programy produktu IBM MQ se od jiných aplikací liší, takže můžete použít stejné vývojové nástroje a také trasovací prostředky produktu IBM MQ .

z/OS Při testování aplikací produktu CICS s produktem IBM MQ for z/OSmůžete použít diagnostický nástroj pro provádění CICS (CEDF). CEDF zadrží vstup a výstup každé volání MQI, stejně jako volání všem službám CICS . V prostředí CICS můžete také napsat ukončovací program pro přechod rozhraní API k poskytnutí diagnostických informací před každým voláním MQI i po něm. Další informace o tom, jak to provést, viz “Použití a zápis aplikací v systému IBM MQ for z/OS” na stránce 851.

IBM i Při testování aplikací produktu IBM i můžete použít standardní ladicí program. Chcete-li tento příkaz spustit, použijte příkaz STRDBG.

Práce s výjimkami a chybami

Musíte zvážit, jak zpracovat zprávy, které nelze doručit, a jak vyřešit chybové situace, které vám ohlásí správce front. U některých sestav musíte nastavit volby sestavy na MQPUT.

Související pojmy

"Aspekty návrhu a výkonu pro aplikace produktu z/OS" na stránce 53
Návrh aplikace je jedním z nejdůležitějších faktorů ovlivňujících výkonnost. Toto téma vám pomůže pochopit některé z aspektů návrhu, které se podílejí na výkonu.

"Vyvíjení aplikací pro IBM MQ" na stránce 5
Můžete vyvíjet aplikace k odesílání a přijímání zpráv a ke správě správců front a souvisejících prostředků. IBM MQ podporuje aplikace napsané v mnoha různých jazycích a rámcích.

"Koncepty vývoje aplikací" na stránce 6
K zápisu aplikací IBM MQ můžete použít volbu procedurálních nebo objektově orientovaných jazyků. Odkazy v tomto tématu použijte pro informace o koncepcích produktu IBM MQ , které jsou užitečné pro vývojáře aplikací.

"Psaní procedurální aplikace pro řazení do fronty" na stránce 689
Prostřednictvím těchto informací můžete získat informace o vytváření aplikací pro řazení do front, připojování a odpojování od správce front, publikování a odběru a otevírání a zavírání objektů.

"Zápis procedurálních aplikací klienta" na stránce 874
Co potřebujete vědět, chcete-li psát klientské aplikace na serveru IBM MQ pomocí procedurálního jazyka.

"Vývoj aplikací produktu .NET" na stránce 514
IBM MQ classes for .NET umožňuje programu zapsaným v programovacím rámci .NET pro připojení k serveru IBM MQ jako IBM MQ MQI client nebo k přímému připojení k serveru IBM MQ .

"Vývoj aplikací C++" na stránce 485
Produkt IBM MQ poskytuje třídy C + + ekvivalentní s objekty IBM MQ a některé další třídy ekvivalentní k datovým typům pole. Poskytuje řadu funkcí, které nejsou prostřednictvím rozhraní MQI k dispozici.

"použitíIBM MQ classes for JMS" na stránce 69
IBM MQ classes for Java Message Service (IBM MQ classes for JMS) je poskytovatel JMS , který je dodáván s IBM MQ. Stejně jako implementace rozhraní definovaných v balíku javax.jms poskytuje produkt IBM MQ classes for JMS dvě sady rozšíření rozhraní API produktu JMS .

"použitíIBM MQ classes for Java" na stránce 308
Použijte IBM MQ v prostředí Java . IBM MQ classes for Java umožňuje aplikaci Java připojit se k IBM MQ jako klient IBM MQ nebo se připojit přímo ke správci front IBM MQ .

"Použití rozhraní Model objektu Component Model (IBM MQ Automation Classes for ActiveX)" na stránce 561
Produkt IBM MQ Automation Classes for ActiveX (MQAX) jsou komponenty ActiveX , které poskytují třídy, které můžete použít ve své aplikaci pro přístup k produktu IBM MQ.

Související informace

IBM MQ Technický přehled

Volba použití produktu IBM MQ classes for Java nebo IBM MQ classes for JMS

Aplikace Java může použít buď produkt IBM MQ classes for Java , nebo IBM MQ classes for JMS pro přístup k prostředkům produktu IBM MQ . Každý přístup má své výhody.

Poznámka: Produkt IBM nebude provádět žádná další vylepšení produktu IBM MQ classes for Java a jsou funkčně stabilizovány na úrovni dodávané v produktu IBM MQ 8.0.

Produkt IBM MQ classes for Java zapouzdřuje rozhraní MQI (Message Queue Interface), nativní rozhraní API produktu IBM MQ a používá stejný model objektu jako jiná objektově orientovaná rozhraní, zatímco

produkt IBM MQ classes for Java Message Service implementuje rozhraní Java Message Service (JMS) produktu Oracle.

Pokud jste obeznámeni s produktem IBM MQ v jiných prostředích než Javaza použití procedurálních nebo objektově orientovaných jazyků, můžete své existující znalosti přenést do prostředí produktu Java pomocí produktu IBM MQ classes for Java. Můžete také využít celou řadu funkcí produktu IBM MQ, které nejsou všechny dostupné v produktu IBM MQ classes for JMS.

Pokud nejste obeznámeni s produktem IBM MQnebo pokud již máte zkušenost s produktem JMS , může být snazší používat dobře známé rozhraní API produktu JMS pro přístup k prostředkům produktu IBM MQ pomocí produktu IBM MQ classes for JMS. JMS je také nedílnou součástí platformy Java Platform, Enterprise Edition (Java EE). Aplikace produktu Java EE mohou asynchronně zpracovávat zprávy prostřednictvím objektů typu message-driven bean (MDB) (MDB). JMS je také standardním mechanizmem produktu Java EE pro interakci s asynchronními systémy zasílání zpráv, jako je například IBM MQ. Každý aplikační server, který je ve shodě s produktem Java EE , musí obsahovat poskytovatele JMS , a proto můžete použít produkt JMS ke komunikaci mezi různými aplikačními servery nebo můžete aplikaci z jednoho poskytovatele služeb JMS portovat bez jakýchkoli změn v aplikaci.

“použitíIBM MQ classes for Java” na stránce 308
Použijte IBM MQ v prostředí Java . IBM MQ classes for Java umožňuje aplikaci Java připojit se k IBM MQ jako klient IBM MQ nebo se připojit přímo ke správci front IBM MQ .

“použitíIBM MQ classes for JMS” na stránce 69
IBM MQ classes for Java Message Service (IBM MQ classes for JMS) je poskytovatel JMS , který je dodáván s IBM MQ. Stejně jako implementace rozhraní definovaných v balíku javax.jms poskytuje produkt IBM MQ classes for JMS dvě sady rozšíření rozhraní API produktu JMS .

Scénáře: WebSphere Application Server s IBM MQ
Scénáře: Profil Liberty produktu WebSphere Application Server s produktem IBM MQ

Metody návrhu pro zprávy

Zvažte aspekty uvedené v těchto informacích, které vám pomohou při návrhu zpráv.

Při použití volání MQI k vložení zprávy do fronty můžete vytvořit zprávu. Jako vstup do volání dodáváte některé řídicí informace v *deskriptoru zpráv* (MQMD) a data, která chcete odeslat do jiného programu. Ve fázi návrhu však musíte vzít v úvahu následující skutečnosti, protože ovlivňují způsob, jakým vytváříte zprávy:

Typ zprávy, která se má použít
Navrhujete jednoduchou aplikaci, ve které můžete odeslat zprávu, pak neprovádět žádnou další akci? Nebo se ptáte na odpověď na otázku? Pokud se ptáte na otázku, můžete zahrnout do deskriptoru zpráv název fronty, na kterou chcete obdržet odpověď.

Chcete, aby vaše požadavky a odpovědi byly synchronní? Z toho vyplývá, že jste nastavili časový limit pro odpověď na odpověď na váš požadavek, a pokud odpověď neobdržíte v tomto období, bude se s ní zacházet jako s chybou.

Nebo byste raději pracovali asynchronně, takže vaše procesy nemusí záviset na výskytu specifických událostí, jako jsou například společné signály časování?

Dalším aspektem je to, zda máte všechny své zprávy v rámci pracovní jednotky.

Přiřazení různých priorit ke zprávám
Každému zprávě můžete přiřadit hodnotu priority a definovat ji tak, aby její zprávy byly udržovaly v pořadí podle priority. Pokud tak učiníte, když jiný program načte zprávu z fronty, vždy obdrží zprávu s nejvyšší prioritou. Pokud fronta neuchovává své zprávy v pořadí priority, program, který načte zprávy z fronty, načte je v pořadí, ve kterém byly přidány do fronty.

Programy mohou také vybrat zprávu pomocí identifikátoru, který správce front přiřadl, když byla zpráva vložena do fronty. Případně můžete generovat vlastní identifikátory pro každou zprávu.

Vliv restartování správce front na zprávy

Správce front při restartování zachová všechny trvalé zprávy a po jejich opětném spuštění je zotavuje ze souborů protokolu produktu IBM MQ . Netrvalé zprávy a dočasné dynamické fronty nejsou zachovány. Všechny zprávy, které nechcete vyřadit, musí být definovány jako trvalé, když jsou vytvořeny. Při zápisu aplikace pro IBM MQ for Windows nebo IBM MQ na systémy UNIX and Linux se ujistěte, že víte, jak byl systém nastaven pro alokaci souboru protokolu, aby se snížilo riziko návrhu aplikace, která se spustí na limity souboru protokolu.

Vzhledem k tomu, že zprávy ve sdílených frontách (dostupné pouze v systému IBM MQ for z/OS) jsou drženy v prostředku CF (coupling facility), přechodné zprávy se zachovají po restartu správce front, pokud je prostředek CF stále k dispozici. Dojde-li k selhání prostředku CF, budou přechodné zprávy ztraceny.

Poskytování informací o sobě příjemci zpráv

Obvykle správce front nastaví ID uživatele, ale toto pole může také nastavit vhodně autorizované aplikace, takže můžete zahrnout své vlastní ID uživatele a další informace, které může přijímající program použít pro účely účtování nebo zabezpečení.

Objem přijímacích front

Pokud může být třeba vložit zprávu do několika front, můžete ji publikovat na téma nebo do rozdělovníku.

Pokud může být třeba vložit zprávu do několika front, můžete ji publikovat na téma.

Selektory a vlastnosti zpráv

Zprávy mohou obsahovat metadata přidružená k nim spolu s informačním obsahem hlavní zprávy. Tyto vlastnosti zpráv mohou být užitečné při poskytování dalších dat.

Tato dodatečná data obsahuje dva aspekty, o kterých je důležité vědět:

- Na vlastnosti se nevztahuje ochrana produktu Advanced Message Security (AMS). Chcete-li použít službu AMS k ochraně dat, dejte ji do informačního obsahu a ne do vlastností zprávy.
- Vlastnosti lze použít k provedení výběru zpráv.

Je důležité si uvědomit, že použití selektorů přeruší standardní konvenci zpráv prvního v prvním výstupu. Vzhledem k tomu, že správce front je optimalizován pro tuto pracovní zátěž, neposkytuje se komplexní selektory z důvodů výkonu. Správce front neukládá indexy vlastností zprávy, proto vyhledání zprávy musí být lineární vyhledávání. Čím hlubší je fronta, tím složitější je selektor, a nižší pravděpodobnost, že selektor odpovídající zprávě bude mít negativní vliv na výkon.

Je-li požadován komplexní výběr, doporučuje se filtrovat zprávy pomocí libovolné aplikace nebo stroje pro zpracování, jako je IBM Integration Bus, do různých míst určení. Alternativně může být užitečné použít hierarchii témat.

Poznámka: Produkt IBM MQ classes for Java nepodporuje použití selektorů, pokud chcete použít selektory, které by měly být provedeny prostřednictvím rozhraní API produktu JMS .

Aspekty návrhu a výkonu aplikací

Existuje řada způsobů, jak může špatný návrh programu ovlivnit výkon. Ty mohou být obtížně detekují, protože se může zdát, že se program může sám o sobě provést, ale ovlivnit výkon jiných úloh. V tomto tématu je vysvětleno několik problémů specifických pro programy provádějící volání produktu IBM MQ .

Zde je několik nápadů, které vám pomohou při návrhu účinných aplikací:

- Navrhněte svou aplikaci tak, aby zpracování probíhá souběžně s časem přemýšlení uživatele:
  - Zobrazí panel a umožní uživateli začít psát, dokud se aplikace stále inicializuje.
  - Získejte data, která potřebujete paralelně, z různých serverů.
- Ponechejte připojení a fronty otevřené, pokud je budete opakovaně používat místo opakovaného otevírání a zavírání, připojování a odpojování.

- Avšak serverová aplikace, která vkládá pouze jednu zprávu, by měla použít MQPUT1.
- Správci front jsou optimalizovány pro zprávy v rozsahu 4 KB až 100 kB. Velmi velké zprávy jsou neefektivní; je pravděpodobně lepší posílat 100 zpráv o velikosti 1 MB, každý než 100 MB zprávy. Velmi malé zprávy jsou také neefektivní. Správce front provádí stejné množství práce pro jednobajtovou zprávu jako pro zprávu o velikosti 4 kB.
- Udržujte zprávy v rámci transakce tak, aby mohly být potvrzeny nebo zálohovány současně.
- Použijte dočasnou volbu pro zprávy, které není třeba obnovit.
- Potřebujete-li odeslat zprávu na určitý počet cílových front, zvažte použití distribučního seznamu.

Vliv délky zprávy

Množství dat ve zprávě může ovlivnit výkon aplikace, která tuto zprávu zpracovává. Chcete-li dosáhnout nejlepšího výkonu z vaší aplikace, odešlete pouze základní data ve zprávě. Například, v požadavku na debetní účet bankovního účtu jsou jediné informace, které může být třeba předat z klienta do serverové aplikace, číslo účtu a výši debetu.

Vliv perzistence zpráv

Trvalé zprávy jsou obvykle protokolovány. Protokolovací zprávy snižují výkon aplikace, takže budou používat trvalé zprávy pouze pro základní data. Pokud mohou být data ve zprávě zahozena v případě zastavení nebo selhání správce front, použijte přechodnou zprávu.

z/OS Operace MQPUT a MQGET pro trvalé zprávy budou blokovány, pokud není k dispozici dostatek prostoru pro žurnál zotavení pro záznam operací. Taková podmínka je uvedena ve zprávě protokolu úlohy správce front zprávami CSQJ110E a CSQJ111A. Ujistěte se, že procesy monitorování jsou na místě, aby se tyto podmínky spravoval a vyhnul.

Hledání konkrétní zprávy

Volání MQGET obvykle načte první zprávu z fronty. Použijete-li v deskriptoru zpráv identifikátory zpráv a korelace (*MsgId* a *CorrelId*) k určení konkrétní zprávy, musí správce front prohledat frontu, dokud nenajde příslušnou zprávu. Použití volání MQGET tímto způsobem ovlivňuje výkon aplikace.

Fronty, které obsahují zprávy o různých délkách

Pokud vaše aplikace nemůže používat zprávy pevné délky, postupně zvětšovat a zmenšovat vyrovnávací paměti tak, aby vyhovovala typické velikosti zprávy. Pokud aplikace vydá volání MQGET, které selže, protože vyrovnávací paměť je příliš malá, vrátí se velikost dat zprávy. Přidejte kód do aplikace tak, aby se velikost vyrovnávací paměti změnila odpovídajícím způsobem, a znovu se vydá volání MQGET.

Poznámka: Pokud nenastavíte atribut **MaxMsgLength** explicitně, standardně se nastaví na 4 MB, což může být velmi neefektivní, pokud se tato hodnota použije k ovlivnění velikosti vyrovnávací paměti aplikace.

Frekvence synchronizačních bodů

Programy, které vydávají velmi velká čísla volání MQPUT nebo MQGET v rámci synchronizačního bodu, aniž by je potvrzovala, mohou způsobit problémy s výkonem. Ovlivněné fronty se mohou zaplňovat zprávami, které jsou momentálně nedostupné, zatímco jiné úlohy mohou čekat na získání těchto zpráv. To má dopad z hlediska úložiště a z hlediska podprocesů, které jsou svázány s úlohami, které se pokouší získat zprávy.

Použití volání MQPUT1

Použijte volání MQPUT1 pouze v případě, že máte jedinou zprávu, která má být vložena do fronty. Chcete-li vložit více než jednu zprávu, použijte volání MQOPEN, za nímž následuje řada volání MQPUT a jedno volání MQCLOSE.

Počet používaných podprocesů

Windows Pro produkt IBM MQ for Windowsmůže aplikace vyžadovat velký počet podprocesů. Každému procesu správce front je přidělen maximální povolený počet podprocesů aplikací.

Aplikace mohou používat příliš mnoho podprocesů. Zvažte, zda aplikace tuto možnost nebere v úvahu a že má provést akce buď k zastavení, nebo k ohlášení tohoto typu výskytu.

Vložit trvalé zprávy pod synchronizačním bodem

Trvalé zprávy by měly být vloženy a měly by být pod synchronizačním bodem. Důvodem je to, že při získávání trvalé zprávy mimo synchronizační bod, pokud dojde k selhání, neexistuje způsob, jak aplikace zjistit, zda byla zpráva z fronty odeslána, či nikoli, a zda, zda byla zpráva dostala, byla také ztracena. Při získávání trvalých zpráv v rámci synchronizačního bodu, pokud se cokoli nezdaří, je transakce odvolána a trvalá zpráva se neztratí, protože je stále ve frontě. Podobně lze při ukládání trvalých zpráv tyto zprávy umístit pod synchronizační bod. Dalším důvodem pro vložení a získání trvalých zpráv pod synchronizačním bodem je skutečnost, že trvalý kód zprávy v produktu IBM MQ je silně optimalizován pro synchronizační bod. Takže vložení a získání trvalých zpráv pod synchronizačním bodem je rychlejší než ukládání a získávání trvalých zpráv mimo synchronizační bod.

Je však rychlejší vkládat a získávat přechodné zprávy mimo synchronizační bod, protože dočasný kód v produktu IBM MQ je optimalizovaný pro neaktuální synchronizační bod. Zavádění a získávání trvalých zpráv se provádí rychlostí diskových jednotek, protože trvalá zpráva je uložena na disk. Avšak vkládání a získávání přechodných zpráv probíhá při rychlostech CPU, protože zde není zahrnutý žádný zápis na disk, a to dokonce ani při použití synchronizačního bodu.

Pokud aplikace získává zprávy a v předstihu neví, zda jsou perzistentní nebo ne, lze použít volbu GMO MQGMO_SYNCPINT_IF_PERSISTENT.

Metody návrhu pro rozšířené aplikace

Při návrhu pokročilejších aplikací existují určité techniky, které byste mohli chtít zvážit například při čekání na zprávy, korelování odpovědí, nastavení a používání informací o kontextu, spouštění aplikací automaticky, generování sestav a odebrání afinit zpráv při použití klastrování.

Pro jednoduchou aplikaci IBM MQ musíte rozhodnout, které objekty produktu IBM MQ použít ve vaší aplikaci a které typy zpráv chcete použít. Pro pokročilejší aplikaci může být vhodné použít některé z technik uvedených v následujících sekcích.

Čekání na zprávy

Program, který obsluhuje frontu, může čekat na zprávy od:

- Čeká se na doručení zprávy, nebo uplynutí zadaného časového intervalu (viz “Čekání na zprávy” na stránce 764).
- z/OS Pouze v systému IBM MQ for z/OS , nastavení signálu tak, aby byl program informován při příchodu zprávy. Další informace viz “signalizace” na stránce 764.
- Vytvoření uživatelské procedury zpětného volání, která má být řízena při doručení zprávy; viz “Asynchronní spotřeba zpráv produktu IBM MQ” na stránce 36.
- Pravidelná volání do fronty za účelem zjištění, zda byla doručena zpráva (*polling*). Tato situace se obvykle nedoporučuje, protože může mít vliv na výkon.

Korelace odpovědí

Když v aplikacích IBM MQ program obdrží zprávu, která požaduje, aby vykonala nějakou práci, program obvykle odešle žadateli jednu nebo více zpráv odpovědí.

Chcete-li žadateli pomoci přidružit tyto odpovědi k původnímu požadavku, aplikace může v deskriptoru každé zprávy nastavit pole *correlation identifier* . Programy potom zkopírují identifikátor zprávy požadavku do pole identifikátoru korelace svých zpráv s odpovědí.

Nastavení a použití kontextových informací

Volba *Informace o kontextu* se používá pro přidružení zpráv k uživateli, který je generoval, a pro identifikaci aplikace, která generovala zprávu. Tyto informace jsou užitečné pro zabezpečení, účtování, auditování a určování problémů.

Při vytváření zprávy můžete zadat volbu, která požaduje, aby správce front přidružoval k vaší zprávě výchozí kontextové informace.

Další informace o používání a nastavení kontextových informací viz “kontext zprávy” na stránce 41.

Automatické spouštění programů IBM MQ

Použijte IBM MQ *triggering* , chcete-li spustit program automaticky, když se zprávy dorazí do fronty.

Můžete nastavit podmínky spouštěče ve frontě tak, aby program začal zpracovávat tuto frontu:

- Pokaždé, když přijde zpráva do fronty
- Kdy dorazí první zpráva do fronty
- Když počet zpráv ve frontě dosáhne předdefinovaného počtu

Další informace o spouštění naleznete v tématu “Spuštění aplikací produktu IBM MQ pomocí spouštěčů” na stránce 829. Spouštěcí impuls je pouze jedním ze způsobů, jak spustit program automaticky. Můžete například spustit program automaticky v časovači pomocí jiných zařízení nežIBM MQ .

Multi V systému Multiplatformymůže produkt IBM MQ definovat objekty služby pro spouštění programů produktu IBM MQ při spuštění správce front; viz Objekty služeb.

Generování sestav produktu IBM MQ

V aplikaci můžete požadovat následující sestavy:

- Sestavy výjimek
- sestavy vypršení platnosti
- Hlášení typu Confirm-on-arrival (COA)
- Zprávy COD (Confirm-on-delivery)
- Sestavy upozornění na pozitivní akce (PAN)
- Negativní upozornění na akce (NAN)

Ty jsou popsány v části “Hlášení zpráv” na stránce 15.

Klastry a afinity zpráv

Než začnete používat klastry s více definicemi pro stejnou frontu, prozkoumejte své aplikace, abyste zjistili, zda existuje nějaká možnost, která vyžaduje výměnu souvisejících zpráv.

V rámci klastru může být zpráva směrována do libovolného správce front, který je hostitelem instance příslušné fronty. Proto může být logika aplikací s afinitními zprávami naštvaná.

Můžete mít například dvě aplikace, které se spoléhají na posloupnost zpráv, které mezi nimi tečou ve formě otázek a odpovědí. Může být důležité, aby byly všechny otázky odeslány do stejného správce front a aby všechny odpovědi byly odeslány zpět do jiného správce front. V této situaci je důležité, aby rutina správy pracovní zátěže neodeslala žádné zprávy do žádného správce front, který se právě stane hostitelem instance příslušné fronty.

Kde je to možné, odstraňte spřízněnosti. Odebrání afinit zpráv zvyšuje dostupnost a rozšiřitelnost aplikací.

Další informace najdete v tématu Práce s spřízněnostmi zpráv.

Aspekty návrhu a výkonu pro aplikace produktu IBM i

Tyto informace vám pomohou pochopit, jak může návrh aplikací, podprocesy a úložiště ovlivnit výkon.

Tyto informace jsou rozděleny do dvou sekcí:

- “Aspekty návrhu aplikací” na stránce 52
- “Specifické problémy s výkonem” na stránce 53

Aspekty návrhu aplikací

Existuje řada způsobů, jak může špatný návrh programu ovlivnit výkon. Tyto problémy mohou být obtížně detekují, protože se může zdát, že se program může provést dobře, a ovlivnit tak výkon jiných úloh. Některé problémy specifické pro programy, které provádějí volání IBM MQ for IBM i , jsou vysvětleny v následujících sekcích.

Další informace o návrhu aplikací viz “Aspekty návrhu pro aplikace produktu IBM MQ” na stránce 43.

Vliv délky zprávy
Ačkoli produkt IBM MQ for IBM i umožňuje pojmovat zprávy až 100 MB dat, množství dat ve zprávě ovlivňuje výkon aplikace, která tuto zprávu zpracovává. Chcete-li dosáhnout nejlepšího výkonu z vaší aplikace, odešlete pouze základní data ve zprávě, například v požadavku na debetní bankovní účet, jediné informace, které může být nutné předat z klienta na serverovou aplikaci, je číslo účtu a částka debetu.

Vliv perzistence zpráv
Trvalé zprávy se žurnálují. Žurnálování zpráv snižuje výkon vaší aplikace, takže používejte trvalé zprávy pouze pro základní data. Pokud mohou být data ve zprávě zahozena v případě zastavení nebo selhání správce front, použijte přechodnou zprávu.

Hledání konkrétní zprávy
Volání MQGET obvykle načte první zprávu z fronty. Použijete-li v deskriptoru zpráv identifikátory zpráv a korelace (*MsgId* a *CorrelId*) k určení konkrétní zprávy, musí správce front frontu prohledávat, dokud tuto zprávu nenajde. Použití volání MQGET tímto způsobem ovlivňuje výkon aplikace.

Fronty, které obsahují zprávy o různých délkách
Jsou-li zprávy ve frontě různé délky, abyste určili velikost zprávy, může aplikace použít volání MQGET s polem *BufferLength* nastaveným na nulu, takže i když se volání nezdaří, vrátí velikost dat zprávy. Aplikace pak může opakovat volání a uvést identifikátor zprávy, která byla měřena při prvním volání, a vyrovnávací paměť o správné velikosti. Pokud však existují jiné aplikace obsluhující stejnou frontu, můžete zjistit, že výkon vaší aplikace je snížen, protože jeho druhé volání MQGET tráví čas hledáním zprávy, kterou v době mezi dvěma voláními načtená jiná aplikace.

Pokud vaše aplikace nemůže používat zprávy s pevnou délkou, je dalším řešením tohoto problému použití volání MQINQ k nalezení maximální velikosti zpráv, které může fronta přijmout, a pak tuto hodnotu použít ve volání MQGET. Maximální velikost zpráv pro frontu je uložena v atributu **MaxMsgLen** ve frontě. Tato metoda by mohla používat velké množství paměti, protože hodnota tohoto atributu fronty může být maximální povolenou hodnotou IBM MQ for IBM i, která může být větší než 2 GB.

Frekvence synchronizačních bodů
Programy, které vydávají četné volání MQPUT v rámci synchronizačního bodu, aniž by je potvrzovala, mohou způsobit problémy s výkonem. Ovlivněné fronty se mohou zaplňovat zprávami, které jsou momentálně nepoužitelné, zatímco jiné úlohy mohou čekat na získání těchto zpráv. Tento problém má důsledky z hlediska úložiště a z hlediska podprocesů svázaných s úlohami, které se pokouší získat zprávy.

Použití volání MQPUT1
Použijte volání MQPUT1 pouze v případě, že máte jedinou zprávu, která má být vložena do fronty. Chcete-li vložit více než jednu zprávu, použijte volání MQOPEN, za nímž následuje řada volání MQPUT a jedno volání MQCLOSE.

Počet používaných podprocesů
Aplikace může vyžadovat mnoho podprocesů. Každému procesu správce front je přidělen maximální povolený počet podprocesů. Jsou-li některé aplikace problematické, může to být způsobeno jejich návrhem použitím příliš mnoha podprocesů. Zvažte, zda aplikace tuto možnost nebere v úvahu a že má provést akce buď k zastavení, nebo k ohlášení tohoto typu výskytu. Maximální počet podprocesů, který produkt IBM i povoluje, je 4 095. Výchozí hodnota je však 64. Produkt IBM MQ zpřístupní své procesy až 63 podprocesů.

Specifické problémy s výkonem

Tento oddíl vysvětluje problémy úložiště a špatné výkonnosti.

Problémy s úložištěm
Obdržíte-li systémovou zprávu `CPF0907. Serious storage condition may exist.`, je možné, že zaplníte prostor přidružený ke správcům front produktu IBM MQ for IBM i .

Je vaše aplikace nebo IBM MQ for IBM i spuštěna pomalu?
Pokud vaše aplikace běží pomalu, může označovat, že se nachází ve smyčce, nebo čeká na prostředek, který není k dispozici. Toto pomalé spouštění může být způsobeno také problémem s výkonem. Možná proto, že váš systém pracuje téměř na hranici své kapacity. Tento typ problému je pravděpodobně nejhorší v době špičkového zatížení systému, obvykle v polovině ráno a v polovině odpoledne. (Pokud se vaše síť vyskytuje u více než jedné časové zóny, může se vám zdát, že vrcholové zatížení systému se vyskytne někdy jindy.)

Pokud zjistíte, že degradace výkonu není závislá na zatížení systému, ale někdy se stane, když je systém na lehkou zátěž, pravděpodobně je to špatně navržený aplikační program, který je pravděpodobně na vině. Tento problém se může projevit jako problém, který se vyskytuje pouze v případě, že k určitým frontám přistupujete.

QTOTJOB a QADLTOTJ jsou systémové hodnoty, které stojí za prozkoumání.

Následující symptomy mohou označovat, že produkt IBM MQ for IBM i běží pomalu:

- Je-li systém pomalu odpovídat na příkazy MQSC.
- Pokud se opakované zobrazení hloubky fronty označuje, znamená to, že je fronta zpracovávána pomalu pro aplikaci, se kterou byste očekávali velké množství aktivity fronty.
- Je trasování produktu IBM MQ spuštěno?

Linux Aplikace rozhraní Linux on POWER Systems - Little Endian

Jelikož produkt Linux on POWER Systems - Little Endian podporuje pouze 64bitové aplikace, v produktu IBM MQ není poskytována žádná podpora pro 32bitové aplikace.

Související pojmy

“Aspekty návrhu pro aplikace produktu IBM MQ” na stránce 43
Když se rozhodnete, jak vaše aplikace mohou využívat výhody platforem a prostředí, které máte k dispozici, musíte se rozhodnout, jak používat funkce nabízené produktem IBM MQ.

z/OS Aspekty návrhu a výkonu pro aplikace produktu z/OS

Návrh aplikace je jedním z nejdůležitějších faktorů ovlivňujících výkonnost. Toto téma vám pomůže pochopit některé z aspektů návrhu, které se podílejí na výkonu.

Existuje řada způsobů, jak může špatný návrh programu ovlivnit výkon. Tyto problémy mohou být obtížně detekují, protože se může zdát, že se program může provést dobře, a ovlivnit tak výkon jiných úloh. Několik problémů specifických pro programy, které provádějí volání MQI, je demonstrováno v následujících sekcích.

Další informace o návrhu aplikací viz “Aspekty návrhu pro aplikace produktu IBM MQ” na stránce 43.

Vliv délky zprávy

Ačkoli produkt IBM MQ for z/OS umožňuje pojmovat zprávy až 100 MB dat, množství dat ve zprávě ovlivňuje výkon aplikace, která tuto zprávu zpracovává. Chcete-li dosáhnout nejlepšího výkonu z vaší aplikace, odešlete pouze základní data ve zprávě. Například, v požadavku na dobropis bankovního účtu jsou jedinými informacemi, které může být třeba předat z klienta do serverové aplikace, číslo účtu a částka k odepsání.

Vliv perzistence zpráv

Trvalé zprávy jsou protokolovány. Protokolovací zprávy snižují výkon aplikace, takže budou používat trvalé zprávy pouze pro základní data. Pokud mohou být data ve zprávě zahozena v případě zastavení nebo selhání správce front, použijte přechodnou zprávu.

Data pro trvalé zprávy jsou zapsána do vyrovnávacích pamětí protokolu. Tyto vyrovnávací paměti jsou zapsány do datových sad protokolu, když:

- Vyskytne se potvrzení
- Zpráva se nachází nebo je uvedena mimo synchronizační bod.
- Vyrovnávací paměti WRTHRSH jsou vyplněny

Zpracování mnoha zpráv v jedné jednotce práce může způsobit menší vstup/výstup, než kdyby byly zprávy zpracovány pro každou jednotku práce, nebo z synchronizačního bodu.

Hledání konkrétní zprávy

Volání MQGET obvykle načte první zprávu z fronty. Pokud použijete identifikátory zpráv a korelace (**MsgId** a **CorrelId**), v deskriptoru zpráv určujete určitou zprávu, správce front prohledá frontu, dokud nenajde příslušnou zprávu. Použití příkazu MQGET tímto způsobem ovlivňuje výkon vaší aplikace, protože při hledání konkrétní zprávy může produkt IBM MQ prohledávat celou frontu.

Atribut fronty produktu **IndexType** můžete použít k určení, že má správce front udržovat index, který lze použít ke zvýšení rychlosti operací MQGET ve frontě. Avšak je zde malé snížení výkonu pro udržování indexu, takže jej můžete vygenerovat pouze v případě, že jej potřebujete použít. Můžete zvolit sestavení indexu identifikátorů zpráv nebo identifikátorů korelace, nebo se rozhodnout nesestavovat index pro fronty, kde jsou zprávy načítány sekvenčně. Snažte se mít mnoho různých klíčových hodnot, ne příliš mnoho se stejnou hodnotou. Například Balance1, Balance2a Balance3, ne tři s Balance. Pro sdílené fronty musíte mít správné **IndexType**. Podrobnosti o atributu fronty **IndexType** viz IndexType.

Chcete-li zabránit tomu, aby se čas restartoval při restartu správce front pomocí indexovaných front, použijte parametr QINDXBLD (NOWAIT) v makru CSQ6SYSP . To umožňuje dokončení restartu správce front, aniž by bylo nutné čekat na dokončení sestavení indexu fronty.

Úplný popis atributu **IndexType** a další atributy objektu naleznete v tématu Atributy objektů.

Fronty, které obsahují zprávy o různých délkách

Získejte zprávu s použitím velikosti vyrovnávací paměti odpovídající očekávané velikosti zprávy. Pokud obdržíte návratový kód označující, že zpráva je příliš dlouhá, získejte větší vyrovnávací paměť. Když dojde k selhání tímto způsobem, vrácená délka dat je velikost nekonvertovaných dat zprávy. Zadáte-li pro volání MQGET hodnotu MQGMO_CONVERT a data se rozšíří během převodu, nemusí se do vyrovnávací paměti vejít vyrovnávací paměť. V takovém případě je třeba velikost vyrovnávací paměti zvětšit.

Zadáte-li příkaz MQGET s nulovou délkou vyrovnávací paměti, vrátí velikost zprávy a aplikace získá vyrovnávací paměť této velikosti a znovu vydá příkaz get. Máte-li více aplikací zpracovávajících frontu, mohla již jiná aplikace tuto zprávu zpracovat, když původní aplikace znovu vydala příkaz get. Pokud občas máte rozsáhlé zprávy, možná budete muset pro tyto zprávy získat velkou vyrovnávací paměť a uvolnit ji po

zpracování zprávy. To by mělo přispět ke snížení problémů s virtuálním úložištěm, pokud všechny aplikace mají velké vyrovnávací paměti.

Pokud vaše aplikace nemůže používat zprávy pevné délky, dalším řešením tohoto problému je použít volání MQINQ k vyhledání maximální velikosti zpráv, které může fronta přijmout, a pak tuto hodnotu použít ve volání MQGET . Maximální velikost zpráv pro frontu je uložena v atributu **MaxMsgL** ve frontě. Tato metoda by mohla použít velké množství paměti, nicméně, protože hodnota **MaxMsgL** může být až 100 MB, maximum povolené produktem IBM MQ for z/OS.

Poznámka: Po vložení velkých zpráv do fronty můžete parametr **MaxMsgL** snížit. Například můžete vložit 100 MB zprávu a pak nastavit **MaxMsgL** na 50 bajtů. To znamená, že je stále možné získat větší zprávy, než je očekávaná aplikace.

Frekvence synchronizačních bodů

Programy, které vydávají mnoho volání MQPUT v rámci synchronizačního bodu, aniž by je potvrzovala, mohou způsobit problémy s výkonem. Ovlivněné fronty se mohou zaplňovat zprávami, které jsou momentálně nepoužitelné, zatímco jiné úlohy mohou čekat na získání těchto zpráv. To má dopad z hlediska úložiště a z hlediska podprocesů svázaných s úlohami, které se pokouší získat zprávy.

Pokud máte více aplikací zpracovávajících frontu, obvykle získáte nejlepší výkon, máte-li jednu z následujících možností:

- 100 krátkých zpráv (méně než 1 KB), nebo
- Jedna zpráva pro větší zprávy (100 kB)

pro každý synchronizační bod. Pokud existuje pouze jedna aplikace zpracovávající frontu, musíte mít více zpráv pro každou jednotku práce.

Můžete omezit počet zpráv, které může úloha získat nebo vložit do jedné jednotky zotavení pomocí atributu správce front produktu **MAXUMSGS** . Informace o tomto atributu najdete v popisu příkazu **ALTER QMGR** v Commands (MQSC) Commands.

Výhody volání MQPUT1

Použijte volání MQPUT1 pouze v případě, že máte jedinou zprávu, která má být vložena do fronty. Chcete-li vložit více než jednu zprávu, použijte volání MQOPEN následované řadou volání MQPUT a jedním voláním funkce MQCLOSE .

Kolik zpráv může správce front obsahovat

Lokální fronty

Počet lokálních zpráv, které může správce front zadržet, je v podstatě velikost sady stránek. Můžete mít až 100 stránek (ačkoli je to doporučená sada stránek 0 a sada stránek 1 jsou určeny pro systémové objekty a fronty). S rozšířeným formátem můžete použít sadu stránek a zvýšit kapacitu sady stránek.

Sdílené fronty

Kapacita pro sdílené fronty závisí na velikosti prostředku Coupling Facility (CF). IBM MQ používá struktury seznamu CF, kde základní paměťové jednotky jsou položky a prvky. Každá zpráva je uložena jako 1 položka a obsahuje více prvků obsahujících přidružená data MQMD a další data zprávy. Počet prvků spotřebovaných jednou zprávou závisí na velikosti zprávy a v případě parametru CFLEVEL (5) se pravidla odlehčování uplatní v době MQPUT. Méně prvků je zapotřebí, když jsou data zprávy odložena do Db2 nebo SMDS. Přístup k datům zpráv je pomalejší, když byla zpráva odložena. Viz Performance Supportpac MP1H pro další porovnání výkonu a režie CPU přidružené k odlehčování zpráv.

Co ovlivňuje výkon

Výkon může znamenat, jak rychle lze zprávy zpracovávat, a také to může znamenat, kolik CPU je potřeba na zprávu.

Co ovlivňuje, jak mohou být zpracovány rychlé zprávy

Pro trvalé zprávy je největším dopadem rychlost datových sad protokolu. Rychlost datových sad protokolu závisí na DASD, na kterých jsou. Je proto třeba dbát na to, aby se data protokolů ukládala na nízké využité svazky, aby se omezilo soupeření. Odkládání protokolů MQ snižuje výkon protokolu v případě, že je na I/O více stránek zapsaných na I/O. Z High Performance Fibre connection (zHPF) má také významný výkon na dobu odezvy I/O, je-li subsystém I/O zaneprázdněn.

Je-li požadavek na získání a vložení zprávy, přístup do fronty je během požadavku na zachování integrity fronty uzamčen. Pro účely plánování považujte frontu za zamknutou pro celý požadavek. Takže pokud čas na vložení je 100 mikrosekund a vy máte více než 10.000 požadavků sekund, může dojít k prodlevám. Možná toho dosáhnete lépe, než je tomu v praxi, ale je to dobré obecné pravidlo. Pro zlepšení výkonu můžete použít různé fronty.

Možné důvody pro to mohou být:

- použití společné fronty odpovědí, kterou každá transakce CICS používá
- každé transakci CICS je přidělena jedinečná odpověď na frontu.
- Tato fronta používá odpověď na frontu pro oblast CICS a všechny transakce v oblasti CICS .

Odpověď závisí na počtu požadavků za sekundu a na době odezvy požadavků.

Mají-li být zprávy načteny ze sady stránek, budou pomalejší v porovnání s tím, kdy se zprávy nacházejí ve fondu vyrovnávacích pamětí. Máte-li více zpráv, než se vejde do fondu vyrovnávacích pamětí, pak se budou rozlít na disk. Proto je třeba zajistit, aby fond vyrovnávacích pamětí byl dostatečně velký pro vaše krátké zprávy. Pokud máte zprávy, které zpracujete o mnoho hodin později, je pravděpodobné, že dojde k odložení na disk, takže byste měli očekávat, že tyto zprávy budou pomalejší, než kdyby se nacházely ve fondu vyrovnávacích pamětí.

V případě sdílené fronty závisí rychlost zpráv na rychlosti prostředku Coupling Facility. Prostředek CF v rámci fyzického procesoru bude pravděpodobně rychlejší než externí prostředek CF. Doba odezvy prostředku CF závisí na tom, jak je prostředek CF vytížen. Například v systémech Hursley, kdy byl prostředek prostředku CF obsazeno 17%, doba odezvy byla 14 mikrosekund. Když byl CF vytížen 95%, doba odezvy byla 45 mikrosekund.

Pokud vaše požadavky MQ používají hodně CPU, může to ovlivnit způsob zpracování rychlých zpráv. Protože je-li logická oblast (LPAR) omezena na CPU, aplikace budou zpožděny čekáním na CPU.

Kolik CPU na zprávu

Obecně platí, že větší zprávy používají více CPU, takže se vyvarujte velkých (x MB) zpráv, je-li to možné.

Při načítání specifických zpráv z front je třeba frontu indexovat, aby mohl správce front přejít přímo na zprávu (a vyhnout se tak možnému skenování celé fronty). Není-li fronta indexována, bude fronta skenována od začátku hledání zprávy. Je-li ve frontě 1000 zpráv, možná bude muset skenovat všech 1000 zpráv. Výsledkem je velké množství zbytečného využití procesoru.

Kanály používající TLS mají další náklady kvůli šifrování zprávy.

V produktu MQ V7 můžete vybrat zprávy podle řetězce selektoru spolu s **CORRELID** nebo **MSGID**. Všechny zprávy musí být vyhledány, takže pokud je ve frontě mnoho zpráv, je to drahé.

Je účinnější, aby aplikace neotvírá příkaz OPEN PUT CLOSE než PUT1 PUT1.

Spouštění v CICS

Je-li rychlost příjmu zpráv pro spuštěnou frontu nízká, je efektivní použít spouštěč jako první. Je-li rychlost příjmu zpráv více než 10 zpráv za sekundu, je efektivnější spustit první transakci, pak nechat transakci zpracovat zprávu a získat další zprávu a tak dále. Pokud zpráva nedorazila do

krátkého období (řekněme mezi 0.1 a 1 sekundu), transakce se ukončí. Při vysoké propustnosti možná budete potřebovat více transakcí běžících ke zpracování zpráv a zabráníte tak vzniku zpráv. Pro každou vyrobenou zprávu spouštěče to vyžaduje vložení a získání spouštěcí zprávy, která v podstatě zdvojnásobí náklady na zprávu.

Počet podporovaných připojení nebo souběžných uživatelů

Každé připojení používá virtuální úložiště v rámci správce front, takže čím více souběžných uživatelů bude využívat více paměti. Pokud potřebujete velmi velký fond vyrovnávacích pamětí a velký počet uživatelů, pak byste mohli být omezeni pro virtuální úložiště a možná budete muset snížit velikost vašich fondů vyrovnávacích pamětí.

Je-li použito zabezpečení, správce front uchovává informace ve správci front po dlouhou dobu. Je ovlivněna velikost virtuálního úložiště, které se používá v rámci správce front.

Produkt **CHINIT** může podporovat až 10 000 připojení. Tato hodnota je omezena virtuálním úložištěm. Pokud připojení používá více paměti, například pomocí TLS, zvyšuje se úložiště na každé připojení, což znamená, že server **CHINIT** může podporovat méně připojení. Pokud zpracovával velké zprávy, bude to vyžadovat více paměti pro vyrovnávací paměti v **CHINIT**, takže server **CHINIT** může podporovat méně zpráv.

Připojení ke vzdálenému správci front jsou efektivnější než připojení klienta. Například každý požadavek klienta MQ vyžaduje dva síťové toky (jeden pro požadavek a jeden pro odezvu). Při použití kanálu ke vzdálenému správci front může být v síti odesláno 50 před tím, než se vrátí odezva. Pokud uvažujete o velké klientské síti, může být efektivnější použít správce front koncentrátoru v rozděleném rámečku a mít jeden kanál přicházející do koncentrátoru a ven z něj.

Další věci ovlivňující výkon

Datová sada protokolu by měla být alespoň 1000 cylindrů ve velikosti. Pokud jsou protokoly menší než tato, aktivita kontrolního bodu může být příliš častá. V zatíženém systému by měl být kontrolní bod obvykle každých 15 minut nebo delší, a to velmi vysoké průnik, může být menší než tento. Když se vyskytne kontrolní bod, jsou skenovány oblasti vyrovnávací paměti a 'staré' zprávy a změněné stránky jsou zapsány na disk. Jsou-li kontrolní body příliš časté, může to mít dopad na výkon. Hodnota LOGLOAD může také ovlivnit frekvenci kontrolních bodů. Pokud správce front abnormálně skončí, pak při restartu může být nutné číst zpět až 3 kontrolní body. Nejlepším intervalem kontrolního bodu je rovnováha mezi aktivitou při provedení kontrolního bodu a množstvím dat protokolu, která bude možná třeba číst při restartu správce front.

Při spouštění kanálu došlo k významné režii. Obvykle je lepší spustit kanál a nechat jej připojené, spíše než časté spouštění a zastavení kanálu.

Související informace

MP1H: IBM MQ for z/OS 9.0 Sestava o výkonu

z/OS Aplikace mostu IMS a IMS v systému IBM MQ for z/OS

Tyto informace vám pomohou při psaní aplikací produktu IMS pomocí produktu IBM MQ.

- Chcete-li použít synchronizační body a volání MQI v aplikacích produktu IMS , prohlédněte si téma “Zápis aplikací produktu IMS pomocí produktu IBM MQ” na stránce 58.
- Chcete-li psát aplikace, které používají most IBM MQ - IMS , přečtěte si téma “Zápis aplikací mostu IMS” na stránce 62.

Následující odkazy použijte k vyhledání dalších informací o aplikacích mostu IMS a IMS v systému IBM MQ for z/OS:

- “Zápis aplikací produktu IMS pomocí produktu IBM MQ” na stránce 58
- “Zápis aplikací mostu IMS” na stránce 62

Související pojmy

“Přehled rozhraní fronty zpráv” na stránce 690
Zde jsou uvedeny informace o komponentách rozhraní MQI (Message Queue Interface).

“Připojování k správci front a odpojování od něj” na stránce 704
Chcete-li používat programovací služby IBM MQ , musí mít program připojení ke správci front. V této části se dozvíte, jak se připojit k správci front a odpojit je od správce front.

“Otevírání a zavírání objektů” na stránce 712
Tyto informace poskytují vhled do otevírání a zavírání objektů IBM MQ .

“Vložení zpráv do fronty” na stránce 723
V této části se dozvíte, jak vložit zprávy do fronty.

“Získávání zpráv z fronty” na stránce 737
Tyto informace použijte k získání informací o získávání zpráv z fronty.

“Inquaning about a nastavení atributů objektu” na stránce 815
Atributy jsou vlastnosti, které definují charakteristiky objektu IBM MQ .

“Potvrzení a zálohování jednotek práce” na stránce 818
Tyto informace popisují, jak potvrdit a zazálohovat všechny zotavitelné operace get a put, které se vyskytly v jednotce práce.

“Spuštění aplikací produktu IBM MQ pomocí spouštěčů” na stránce 829
Informace o spouštěčích a o tom, jak spustit aplikace IBM MQ pomocí spouštěčů.

“Práce s MQI a klastry” na stránce 847
Existují speciální volby na voláních a návratových kódech, které se vztahují ke klastrování.

“Použití a zápis aplikací v systému IBM MQ for z/OS” na stránce 851
Aplikace produktu IBM MQ for z/OS mohou být vytvořeny z programů spuštěných v mnoha různých prostředích. To znamená, že mohou využít výhody zařízení dostupných ve více než jednom prostředí.

Zápis aplikací produktu IMS pomocí produktu IBM MQ

Při použití produktu IBM MQ v aplikacích produktu IMS jsou k dispozici další aspekty, které zahrnují volání rozhraní API produktu MQ a mechanizmus používaný pro synchronizační bod.

Následující odkazy použijte k vyhledání dalších informací o zápisu aplikací IMS v systému IBM MQ for z/OS:

- “Synchronizační body v aplikacích produktu IMS” na stránce 58
- “Volání MQI v aplikacích produktu IMS” na stránce 59

Omezení

Existují omezení, ve kterých volání rozhraní API produktu IBM MQ může používat aplikace s použitím adaptéru IMS .

Následující volání rozhraní API produktu IBM MQ nejsou v rámci aplikace s použitím adaptéru IMS podporována:

- MQCB
- FUNKCE MQCB_
- MQCTL

Související pojmy

“Zápis aplikací mostu IMS” na stránce 62
Toto téma obsahuje informace o zápisu aplikací pro použití mostu IBM MQ - IMS .

Synchronizační body v aplikacích produktu IMS

V aplikaci IMS zavedete synchronizační bod pomocí volání příkazu IMS , jako je GU (get unique) na IOPCB a CHKP (checkpoint).

Chcete-li odvolat všechny změny od předchozího kontrolního bodu, můžete použít volání IMS ROLB (rollback). Další informace najdete v následující dokumentaci:

- IMS 13 Application Programming APG SC19-3646
- IMS 13 Rozhraní API pro programování aplikací APR SC19-3647

Správce front je účastníkem protokolu s dvoufázovým potvrzováním; správce synchronizačního bodu IMS je koordinátorem.

Všechny otevřené manipulátory jsou uzavřeny adaptérem IMS v synchronizačním bodu (s výjimkou dávkového nebo neřízeného prostředí BMP). Důvodem je skutečnost, že jiný uživatel by mohl zahájit další jednotku práce a kontrola zabezpečení produktu IBM MQ se provádí při volání MQCONN, MQCONNX a MQOPEN, nikoli při volání MQPUT nebo MQGET.

V prostředí WFI (Wait-for-Input) nebo PWFI (pseudo Wait-for-Input) IMS však IBM MQ neuzavřete obslužné rutiny, dokud nebude doručena další zpráva nebo dokud nebude do aplikace vrácen stavový kód QC. Pokud aplikace čeká v oblasti IMS a některá z těchto obslužných rutin patří ke spouštěcím frontám, spuštění se neproběhne, protože fronty jsou otevřené. Z tohoto důvodu by aplikace spuštěné v prostředí WFI nebo PWFI měly explicitně MQCLOSE před provedením příkazu GU na port IOPCB pro další zprávu.

Je-li při volání MQDISC použita aplikace IMS (buď BMP nebo MPP), otevřené fronty jsou zavřeny, ale není proveden žádný implicitní synchronizační bod. Pokud aplikace skončí normálně, všechny otevřené fronty se zavřou a dojde k implicitnímu odevzdání. Pokud se aplikace ukončí nestandardně, všechny otevřené fronty se zavřou a dojde k implicitnímu odvolání.

Volání MQI v aplikacích produktu IMS

Tyto informace použijte k seznámení s použitím volání MQI v aplikacích serveru a v aplikacích Poptávka.

Tento oddíl pokrývá použití volání MQI v následujících typech aplikací produktu IMS :

- “Serverové aplikace” na stránce 59
- “Dotazové aplikace” na stránce 61

Serverové aplikace

Zde je obrys modelu aplikace serveru MQI:

```
Initialize/Connect
.
Open queue for input shared
.
Get message from IBM MQ queue
.
Do while Get does not fail
.
If expected message received
Process the message
Else
Process unexpected message
End if
.
Commit
.
Get next message from IBM MQ queue
.
End do
.
Close queue/Disconnect
.
END
```

Ukázkový program CSQ4ICB3 zobrazuje implementaci ve formátu BMP pomocí tohoto modelu v produktu C/370. Program nejprve naváže komunikaci s IMS a potom pomocí IBM MQ:

```
main()
----
Call InitIMS
```

```
If IMS initialization successful
Call InitMQM
If IBM MQ initialization successful
Call ProcessRequests
Call EndMQM
End-if
End-if

Return
```

Inicializace IMS určuje, zda byl program volán jako řízený zprávami nebo jako dávkový soubor typu BMP orientovaný na dávky a řídí příslušným způsobem připojení správce front IBM MQ a příslušné fronty:

```


InitIMS
-------
Get the IO, Alternate and Database PCBs
Set MessageOriented to true

Call ctdli to handle status codes rather than abend
If call is successful (status code is zero)
While status code is zero
Call ctdli to get next message from IMS message queue
If message received
Do nothing
Else if no IOPBC
Set MessageOriented to false
Initialize error message
Build 'Started as batch oriented BMP' message
Call ReportCallError to output the message
End-if
Else if response is not 'no message available'
Initialize error message
Build 'GU failed' message
Call ReportCallError to output the message
Set return code to error
End-if
End-if
End-while
Else
Initialize error message
Build 'INIT failed' message
Call ReportCallError to output the message
Set return code to error
End-if

Return to calling function
```

Inicializace IBM MQ se připojí ke správci front a otevře fronty. V souboru BMP s řízeným zprávami se volá po každém synchronním synchronizačním bodu IMS ; v souboru BMP s dávkovým zaměřením se toto volání volá pouze během spouštění programu:

```


InitMQM
-------
Connect to the queue manager
If connect is successful
Initialize variables for the open call
Open the request queue
If open is not successful
Initialize error message
Build 'open failed' message
Call ReportCallError to output the message
Set return code to error
End-if
Else
Initialize error message
Build 'connect failed' message
Call ReportCallError to output the message
Set return code to error
End-if

Return to calling function
```

Implementace modelu serveru v MPP je ovlivněna skutečností, že MPP zpracovává jedinou jednotku práce na vyvolání. Důvodem je to, že když je proveden synchronizační bod (GU), jsou uzavřeny obslužné rutiny

připojení a fronty a je doručena další zpráva IMS . Toto omezení může být částečně překonán jedním z následujících způsobů:

- **Zpracování mnoha zpráv v rámci jedné jednotky-práce**

  To zahrnuje:

  – Čtení zprávy
  – Zpracování požadovaných aktualizací
  – Zadání odpovědi

  ve smyčce, dokud nebudou zpracovány všechny zprávy nebo dokud nezpracuje nastavený maximální počet zpráv, v tomto okamžiku se provede synchronizační bod.

  Touto cestou lze přistupovat pouze k určitým typům aplikací (například k aktualizaci jednoduché databáze nebo dotazu). Ačkoli lze zprávy s odpovědí rozhraní MQI odesílat s oprávněním původce zprávy modulu MQI, která je zpracovávána, je třeba pečlivě řešit důsledky zabezpečení pro všechny aktualizace prostředků produktu IMS .
- **Zpracovává se jedna zpráva na vyvolání MPP a zajišťuje více plánování MPP pro zpracování všech dostupných zpráv.**

  Pomocí programu monitoru spouštěčů produktu IBM MQ IMS (CSQQTRMN) můžete naplánovat transakci MPP, pokud ve frontě IBM MQ jsou zprávy a žádné aplikace, které ji obsluhují.

  Pokud monitor spouštěčů spustí MPP, jméno správce front a jméno fronty jsou předány programu, jak je zobrazeno v následujícím extraktu kódu COBOL:

```
* Data definition extract
01 WS-INPUT-MSG.
05 IN-LL1         PIC S9(3) COMP.
05 IN-ZZ1         PIC S9(3) COMP.
05 WS-STRINGPARM      PIC X(1000).
01 TRIGGER-MESSAGE.
COPY CMQTMC2L.
*
* Code extract
GU-IOPCB SECTION.
MOVE SPACES TO WS-STRINGPARM.
CALL 'CBLTDLI' USING GU,
IOPCB,
WS-INPUT-MSG.
IF IOPCB-STATUS = SPACES
MOVE WS-STRINGPARM TO MQTMC.
*  ELSE handle error
*
* Now use the queue manager and queue names passed
DISPLAY 'MQTMC-QMGRNAME     ='
MQTMC-QMGRNAME OF MQTMC '='.
DISPLAY 'MQTMC-QNAME       ='
MQTMC-QNAME  OF MQTMC '='.
```

Model serveru, který se očekává, že bude dlouhodobě spuštěnou úlohou, je lépe podporován v oblasti dávkového zpracování, ačkoli BMP nelze spustit pomocí CSQQTRMN.

Dotazové aplikace

Typická aplikace IBM MQ zahajující dotaz nebo aktualizace pracuje následujícím způsobem:

- Shromáždit data od uživatele
- Vložit jednu nebo více zpráv produktu IBM MQ
- Získejte zprávy odpovědi (možná budete muset na ně počkat)
- Zadejte odpověď uživateli.

Vzhledem k tomu, že zprávy vkládané do front produktu IBM MQ nejsou zpřístupněny ostatním aplikacím produktu IBM MQ , dokud nejsou potvrzeny, musí být buď uvedeny v synchronizačním bodu, nebo musí být aplikace IMS rozdělena do dvou transakcí.

Pokud dotaz zahrnuje vložení jedné zprávy, můžete použít volbu *bez synchronizačního bodu* ; pokud je však dotaz složitější nebo pokud jsou zahrnuty aktualizace prostředků, může dojít k problémům s konzistencí, dojde-li k selhání a nepoužijete syncpointing.

Chcete-li to překonat, můžete pomocí volání MQI rozdělit transakce MPP IMS pomocí volání MQI, viz téma *IMS/ESA Application Programming: Data Communication* , kde získáte informace o tomto tématu. To umožňuje implementovat dotazovací program v MPP:

```
Initialize first program/Connect
.
Open queue for output
.
Put inquiry to IBM MQ queue
.
Switch to second IBM MQ program, passing necessary data in save
pack area (this commits the put)
.
END
.
.
Initialize second program/Connect
.
Open queue for input shared
.
Get results of inquiry from IBM MQ queue
.
Return results to originator
.
END
```

Zápis aplikací mostu IMS

Toto téma obsahuje informace o zápisu aplikací pro použití mostu IBM MQ - IMS .

Informace o mostu IBM MQ - IMS naleznete v tématu Most systému IMS.

Následující odkazy použijte k vyhledání více informací o zápisu aplikací mostu IMS v systému IBM MQ for z/OS:

- “Jak se most IMS zabývá zprávami” na stránce 62
- “Zápis transakčních programů IMS prostřednictvím IBM MQ” na stránce 873

Související pojmy

“Zápis aplikací produktu IMS pomocí produktu IBM MQ” na stránce 58
Při použití produktu IBM MQ v aplikacích produktu IMS jsou k dispozici další aspekty, které zahrnují volání rozhraní API produktu MQ a mechanizmus používaný pro synchronizační bod.

Jak se most IMS zabývá zprávami

Pokud použijete most IBM MQ - IMS k odesílání zpráv do aplikace produktu IMS , je třeba vytvořit zprávy ve speciálním formátu.

Musíte také vložit zprávy do front produktu IBM MQ , které byly definovány s třídou úložiště, která určuje skupinu XCF a název člena cílového systému IMS . Jsou známy jako fronty mostu MQ-IMS nebo pouze fronty **mostu** .

Most IBM MQ-IMS vyžaduje výlučný vstupní přístup (MQOO_INPUT_EXCLUSIVE) do fronty mostu, je-li definován s QSGDISP (QMGR) nebo pokud je definován s QSGDISP (SHARED) spolu s volbou NOSHARE.

Uživatel se nemusí přihlašovat k produktu IMS před odesláním zpráv do aplikace IMS . ID uživatele v poli *UserIdentifier* struktury MQMD se používá pro kontrolu zabezpečení. Úroveň kontroly se určuje, když se produkt IBM MQ připojí k produktu IMSa je popsán v tématu Řízení přístupu k aplikacím pro most IMS. To umožní implementaci pseudo přihlášení.

Most IBM MQ - IMS přijímá následující typy zpráv:

- Zprávy obsahující transakční data produktu IMS a strukturu MQIIH (popsané v části MQIIH):

```
MQIIH LLZZ<trancode><data>[LLZZ<data>][LLZZ<data>]
```

Poznámka:

1. Hranaté závorky, [], představují volitelné vícesegmentové segmenty.
2. Chcete-li použít strukturu MQIIH, nastavte pole *Format* struktury MQMD na MQFMT_IMS.

- Zprávy obsahující transakční data produktu IMS , ale ne strukturu MQIIH:

```
LLZZ<trancode><data> \
[LLZZ<data>][LLZZ<data>]
```

IBM MQ ověřuje data zprávy, aby se zajistilo, že součet bajtů LL plus délka MQIIH (je-li přítomna) se rovná délce zprávy.

Pokud most produktu IBM MQ - IMS získává zprávy z front mostu, zpracuje je následujícím způsobem:

- Obsahuje-li zpráva strukturu MQIIH, most ověřuje záhlaví MQIIH (viz MQIIH), sestaví záhlaví OTMA a odešle zprávu do produktu IMS. Kód transakce je uveden ve vstupní zprávě. Pokud se jedná o LTERM, IMS odpoví zprávou DFS1288E . Pokud kód transakce představuje příkaz, příkaz IMS provede tento příkaz; v opačném případě je zpráva zařazena do fronty v produktu IMS pro danou transakci.
- Pokud zpráva obsahuje transakční data produktu IMS , ale ne strukturu MQIIH, most IMS provede následující předpoklady:
  - Kód transakce je v bajtech 5 až 12 uživatelských dat.
  - Transakce se nachází v nekonverzačním režimu.
  - Transakce je v režimu vázaného zpracování 0 (operace commit-then-send)
  - *Format* v MQMD se používá jako *MFSMapName* (na vstupu).
  - Režim zabezpečení je MQISS_CHECK.

  Zpráva odpovědi je také sestavena bez struktury MQIIH a přebírá *Format* pro deskriptor MQMD z výstupu *MFSMapName* výstupu IMS .

Most IBM MQ - IMS používá jeden nebo dva Tpipes pro každou frontu IBM MQ :

- Synchronizovaná Tpipe se používá pro všechny zprávy pomocí režimu vázaného zpracování 0 (COMMIT_THEN_SEND) (tyto zobrazují se SYN ve stavovém poli u příkazu TPIPE xxxx produktu IMS /DIS TMEMBER)
- Nesynchronizovaná Tpipe se používá pro všechny zprávy pomocí režimu vázaného zpracování 1 (SEND_THEN_COMMIT)

Tpipe jsou vytvářeny produktem IBM MQ při jejich prvním použití. Nesynchronizovaná Tpipe existuje, dokud se produkt IMS nerestartuje. Synchronizované nástroje Tpipe existují, dokud produkt IMS nestuduje. Tyto Trubury nelze odstranit sami.

Další informace o tom, jak produkt IBM MQ - IMS pracuje se zprávami, naleznete v následujících tématech:

- “Mapování zpráv produktu IBM MQ na typy transakcí produktu IMS” na stránce 64
- “Pokud zprávu nelze vložit do fronty IMS” na stránce 64
- “Kódy zpětné vazby mostu IMS” na stránce 65
- “Pole MQMD ve zprávách z mostu IMS” na stránce 65
- “Pole MQIIH ve zprávách z mostu IMS” na stránce 66
- “Odpověď na zprávy z IMS” na stránce 67
- “Použití PCB s alternativním obsahem v transakcích produktu IMS” na stránce 67
- “Odesílání nevyžádaných zpráv z produktu IMS” na stránce 67

- “Segmentace zpráv” na stránce 68
- “Převod dat” na stránce 68

Související pojmy

“Zápis transakčních programů IMS prostřednictvím IBM MQ” na stránce 873
Kódování vyžadované pro zpracování transakcí produktu IMS prostřednictvím produktu IBM MQ závisí na formátu zprávy, který je vyžadován transakcí IMS a o rozsahu odpovědí, které může vrátit. Existuje však několik bodů, které byste měli zvážit, když vaše aplikace zpracovává informace o formátování obrazovky produktu IMS .

Mapování zpráv produktu IBM MQ na typy transakcí produktu IMS
Tabulka popisující mapování zpráv produktu IBM MQ na typy transakcí produktu IMS .

Tabulka 4. Mapování zpráv produktu IBM MQ na typy transakcí produktu IMS

IBM MQ typ zprávy	**Potvrzení-then-send (režim 0)- používá synchronizované IMS Tpipe**	**Odeslat-pak-potvrdit (režim 1)- používá nesynchronizované IMS Tpipe**
Trvalé zprávy IBM MQ	• Obnovitelné transakce plné funkce • Nezotavitelné transakce jsou odmítnuty IMS	• Transakce Fastpath • Konverzační transakce • Transakce s plnou funkcí
Netrvalé zprávy IBM MQ	• Nezotavitelné transakce plné funkce • Napravitelné transakce jsou povoleny s IMS V8 a APAR PQ61404 a všemi pozdějšími verzemi IMS	• Transakce Fastpath • Konverzační transakce • Transakce s plnou funkcí

Poznámka: Příkazy IMS nemohou používat trvalé zprávy IBM MQ s režimem vázaného zpracování 0. Další informace naleznete v příručce *IMS/ESA Open Transaction Manager Access User's Guide* .

Pokud zprávu nelze vložit do fronty IMS
Informace o akcích, které mají být provedeny v případě, že zprávu nelze vložit do fronty produktu IMS .

Pokud zprávu nelze vložit do fronty IMS , provede příkaz IBM MQnásledující akce:

- Pokud nelze zprávu vložit do IMS , protože zpráva je neplatná, zpráva se umístí do fronty nedoručených zpráv a zpráva se odešle na systémovou konzolu.
- Je-li zpráva platná, ale je odmítnuta produktem IMS, IBM MQ pošle na systémovou konzolu chybovou zprávu, zpráva obsahuje chybový kód IMS a zpráva IBM MQ se umístí do fronty nedoručených zpráv. Pokud má chybový kód IMS hodnotu 001A, IMS pošle zprávu IBM MQ obsahující příčinu selhání odpovědi na frontu pro odpověď.

  Poznámka: Za výše uvedených okolností, pokud IBM MQ nemůže z nějakého důvodu vložit zprávu do fronty nedoručených zpráv, je zpráva vrácena do původní fronty produktu IBM MQ . Do systémové konzoly se odešle chybová zpráva a z této fronty se neodesílají žádné další zprávy.

  Chcete-li zprávy znovu odeslat, proveďte ***jednu*** z následujících možností:

  – Zastavte a restartujte Tpipe v IMS odpovídající frontě
  – Změňte frontu na GET (DISABLED), a znovu na GET (ENABLED)
  – Zastavte a znovu spusťte IMS nebo OTMA
  – Zastavte a znovu spusťte svůj subsystém IBM MQ .
- Je-li zpráva odmítnuta IMS pro cokoli jiného, než je chyba zprávy, vrátí se zpráva IBM MQ do původní fronty, IBM MQ zastaví zpracování fronty a chybová zpráva se odešle na systémovou konzolu.

Je-li vyžadována zpráva o výjimce, most ji vloží do fronty pro odpověď s oprávněním původce. Pokud zpráva nemůže být vložena do fronty, zpráva sestavy se umístí do fronty nedoručených zpráv s oprávněním mostu. Pokud ji nelze umístit do fronty DLQ, je vyřazena.

Kódy zpětné vazby mostu IMS
Kódy příčiny IMS jsou obvykle výstupem v hexadecimálním formátu ve zprávách konzoly produktu IBM MQ , jako je například CSQ2001I (například, chybový kód 0x001F). IBM MQ kódů zpětné vazby, jak je vidět v záhlaví zpráv vložených do fronty nedoručených zpráv, jsou desítková čísla.

Kódy zpětné vazby mostu IMS jsou v rozsahu 301 až 399, nebo 600 až 855 pro chybový kód NACK 0x001A. Jsou namapovány z chybových kódů IMS-OTMA takto:

1. Zjištěný kód IMS-OTMA se převede z hexadecimálního čísla na desítkové číslo.
2. Do čísla, které je výsledkem výpočtu v 1, je přidáno 300, což dává kód IBM MQ *Feedback* .
3. Smysl IMS-OTMA 0x001A, desetinné číslo 26 je speciální případ. Je vygenerován kód *Feedback* v rozsahu 600-855.
   a. Kód příčiny IMS-OTMA se převede z hexadecimálního čísla na desítkové číslo.
   b. 600 je přidáno k číslu, které je výsledkem výpočtu v a, což poskytuje kód IBM MQ *Feedback* .

Informace o chybových kódech služby IMS-OTMA najdete v tématu Chybové kódy OTMA pro zprávy NAK.

Pole MQMD ve zprávách z mostu IMS
Informace o polích MQMD ve zprávách z mostu IMS .

MQMD z původní zprávy je přenášeno IMS v sekci Uživatelská data záhlaví OTMA. Pokud zpráva pochází z produktu IMS, je tato zpráva vytvořena uživatelskou procedurou rozpoznání místa určení IMS . MQMD zprávy přijaté od produktu IMS je sestaveno takto:

StrucID
"MD"

Verze
MQMD_VERSION_1

Sestava
MQRO_NONE

MsgType
MQMT_REPLY

Vypršení
Je-li hodnota MQIIH_PASS_EXPIRATION nastavena v poli Příznaky MQIIH, obsahuje toto pole zbývající čas vypršení platnosti, jinak je nastaven na hodnotu MQEI_UNLIMITED.

Zpětná vazba
MQFB_NONE

Kódování
MQENC.Native (kódování systému z/OS)

CodedCharSetId
MQCCSI_Q_MGR (CodedCharSetID) systému z/OS)

Formát
MQFMT_IMS, je-li MQMD.Format vstupní zprávy je MQFMT_IMS, jinak IOPCB.MODNAME

Priorita
MQMD.Priority vstupní zprávy

Trvání
Závisí na režimu vázaného zpracování: MQMD.Persistence vstupní zprávy, pokud perzistence CM-1; odpovídá zotavitelnosti zprávy IMS , pokud CM-0

MsgId
MQMD.MsgId , pokud MQRO_PASS_MSG_ID, jinak nové MsgId (výchozí)

CorrelId
MQMD.CorrelId ze vstupní zprávy, je-li MQRO_PASS_CORREL_ID, jinak MQMD.MsgId ze vstupní zprávy (předvolba)

BackoutCount
0

ReplyToQ
Mezery

ReplyToQMgr
Mezery (nastavení na lokální název správce front qmgr správcem front během operace MQPUT)

UserIdentifier
MQMD.UserIdentifier vstupní zprávy

AccountingToken
MQMD.AccountingToken vstupní zprávy.

ApplIdentityData
MQMD.ApplIdentityData vstupní zprávy

PutApplType
MQAT_XCF, pokud není žádná chyba, jinak MQAT_BRIDGE

PutApplName
<XCFgroupName> <XCFmemberName> není-li žádná chyba, jinak název QMGR

PutDate
Datum, kdy byla zpráva vložena

PutTime
Čas, kdy byla zpráva vložena

ApplOriginData
Mezery

Pole MQIIH ve zprávách z mostu IMS
Získejte informace o polích MQIIH ve zprávách z mostu IMS .

MQIIH zprávy přijaté z produktu IMS je sestaveno takto:

StrucId
"IIH"

Verze
1

StrucLength
84

Kódování
MQENC_NATIVE

CodedCharSetId
MQCCSI_Q_MGR

Formát
MQIIH.ReplyToFormat vstupní zprávy, pokud MQIIH.ReplyToFormat není prázdný, jinak IOPCB.MODNAME

Příznaky
0

LTermOverride
Název LTERM (Tpipe) z hlavičky OTMA

MFSMapName
Název mapování z záhlaví OTMA

Formát ReplyTo
Mezery

Ověřovatel
MQIIH.Authenticator vstupní zprávy, je-li zpráva odpovědi vložena do fronty mostu MQ-IMS , jinak prázdné.

ID TranInstance
Konverzační ID/token serveru z hlavičky OTMA, pokud se jedná o konverzaci. Ve verzích systému IMS starších než V14je toto pole vždy prázdné, pokud se nejedná o konverzaci. V systému IMS V14 může být toto pole nastaveno systémem IMS i v případě, že se nejedná o konverzaci.

TranState
"C", je-li v konverzaci, jinak prázdné

CommitMode
Režim vázaného zpracování v záhlaví OTMA ("0" nebo "1")

SecurityScope
Prázdný

Vyhrazené
Prázdný

Odpověď na zprávy z IMS
Když je transakce systému IMS transakcí ISRTs na její IPCB, zpráva je směrována zpět na původní LTERM nebo TPIPE.

Tyto zprávy jsou v produktu IBM MQ zobrazeny jako zprávy odpovědí. Zprávy odpovědí z produktu IMS jsou vloženy do fronty pro odpovědi určené v původní zprávě. Pokud zpráva nemůže být vložena do fronty pro odpověď, je umístěna do fronty nedoručených zpráv s použitím autority mostu. Pokud zprávu nelze umístit do fronty nedoručených zpráv, odešle se do produktu IMS negativní potvrzení, které potvrdí, že zprávu nelze přijmout. Odpovědnost za zprávu se pak vrátí do IMS. Pokud používáte režim vázaného zpracování 0, zprávy z této Tpipe se neodesílají na most a zůstanou ve frontě IMS . To znamená, že se neodešlou žádné další zprávy, dokud nebude restartován. Používáte-li režim vázaného zpracování 1, může pokračovat jiná práce.

Pokud má odpověď strukturu MQIIH, je její typ formátu MQFMT_IMS; pokud ne, je tento typ formátu určen pomocí názvu IMS MOD použitého při vkládání zprávy.

Použití PCB s alternativním obsahem v transakcích produktu IMS
Když transakce IMS používá PCB s alternativním odezvou (ISRTs na ALTPCB nebo vydává volání CHNG na modifikovatelný PCB), vyvolá se uživatelská procedura před směrováním (DFSYPRX0), aby určila, zda by zpráva měla být přesměrována.

Má-li být zpráva přesměrována, je pro potvrzení cíle vyvolána uživatelská procedura rozpoznání místa určení (DFSYDRU0) a informace o hlavičce uvádí téma Použití uživatelských procedur OTMA v produktu IMS a DFSYPRX0 pro informace o těchto ukončovacích programech.

Pokud se akce neprovedou ve výstupních procedurách, veškerý výstup z transakcí produktu IMS zahájených ze správce front produktu IBM MQ bez ohledu na to, zda má být IOPCB nebo ALTPCB, bude vrácen do stejného správce front.

Odesílání nevyžádaných zpráv z produktu IMS
Chcete-li odesílat zprávy z produktu IMS do fronty produktu IBM MQ , je třeba vyvolat transakci IMS , která má hodnotu ISRTs na ALTPCB.

You need to write pre-routing and destination resolution exits to route unsolicited messages from IMS and build the OTMA user data, so that the MQMD of the message can be built correctly. Informace o těchto ukončovacích programech najdete v tématu Ukončení předběžného směrování DFSYPRX0 a Uživatelská procedura rozpoznání místa určení .

Poznámka: Most IBM MQ - IMS neví, zda je zpráva, kterou obdrží, odpověď nebo nevyžádaná zpráva. To zpracovává stejnou zprávu v každém případě, sestavení MQMD a MQIIH odpovědi založené na OTMA UserData , která byla doručena se zprávou.

Nevyžádané zprávy mohou vytvářet nové Tpipe. Například pokud se existující transakce IMS přepnula na nový LTERM (například PRINT01), ale implementace vyžaduje, aby byl výstup dodán

prostřednictvím OTMA, vytvoří se nové Tpipe (s názvem PRINT01 v tomto příkladu). Standardně se jedná o nesynchronizovanou Tpipe. Vyžaduje-li implementace zprávu, která má být obnovitelná, nastavte výstupní příznak ukončení rozlišení místa určení. Další informace viz příručka *IMS Customization Guide* .

Segmentace zpráv
Můžete definovat transakce IMS jako očekávaný vstup z jednoho nebo více segmentů.

Původní aplikace IBM MQ musí vytvořit uživatelský vstup za strukturu MQIIH jako jeden nebo více segmentů dat LLZZ. Všechny segmenty zprávy IMS musí být obsaženy v jediné zprávě IBM MQ odeslané s jedním MQPUT.

Maximální délka segmentu LLZZ-datového segmentu je definována systémem IMS/OTMA (32767 bajtů). Celková délka zprávy IBM MQ je součtem bajtů LL a délky struktury MQIIH.

Všechny segmenty odpovědi jsou obsaženy v jediné zprávě IBM MQ .

Pro zprávy ve formátu MQFMT_IMS_VAR_STRING existuje další omezení pro omezení velikosti 32 kB. Když se data ve zprávě ASCII-smíšená CCSID převedou na zprávu CCSID se smíšeným EBCDIC, je bajt shift-in nebo shift-out bajt přidán pokaždé, když dojde k přechodu mezi SBCS a znaky DBCS. Omezení 32 KB se použije na maximální velikost zprávy. To znamená, že pole LL ve zprávě nesmí přesáhnout 32 kB, zpráva nesmí překročit 32 kB včetně všech znaků shift-in a shift-out. Verze aplikace, kterou tato zpráva musí mít, musí umožňovat toto.

Převod dat
Převod dat provádí buď prostředek distribuovaného systému front (který může volat všechny nezbytné uživatelské procedury), nebo agent fronty v rámci skupiny (který nepodporuje použití uživatelských procedur) při vložení zprávy do cílové fronty, která má informace o XCF definované pro příslušnou paměťovou třídu. Konverze dat se nevyskytne, když je zpráva doručena do fronty prostřednictvím publikování/odběru.

Všechny potřebné uživatelské procedury musí být k dispozici pro prostředek distribuovaných front v datové sadě, na kterou odkazuje příkaz CSQXLIB DD. To znamená, že můžete odesílat zprávy do aplikace IMS pomocí mostu IBM MQ - IMS z libovolné platformy IBM MQ .

Pokud došlo k chybám převodu, je zpráva vložena do fronty bez převodu; výsledkem je nakonec, že most produktu IBM MQ - IMS je považován za chybu, protože most nemůže rozpoznat formát záhlaví. Dojde-li k chybě konverze, odešle se na konzolu z/OS chybová zpráva.

Podrobné informace o převodu dat obecně najdete v příručce “Zápis uživatelských procedur pro převod dat” na stránce 948 .

Odesílání zpráv na most IBM MQ - IMS

Chcete-li zajistit, aby byl převod proveden správně, musíte správci front sdělit, jaký má formát zprávy.

Pokud má zpráva strukturu MQIIH, musí být *Format* v MQMD nastaveno na vestavěný formát MQFMT_IMS a produkt *Format* v MQIIH musí být nastaven na název formátu, který popisuje data zprávy. Pokud zde není MQIIH, nastavte *Format* v deskriptoru MQMD na název vašeho formátu.

Jsou-li vaše data (jiná než LLZs) všechna znaková data (MQCHAR), použijte jako název vašeho formátu (v záhlaví MQIIH nebo MQMD) vestavěný formát MQFMT_IMS_VAR_STRING. Jinak použijte jméno vašeho formátu, v takovém případě musíte také poskytnout uživatelskou proceduru pro převod dat pro svůj formát. Výjezd musí obsloužit konverzi LLZs ve vaší zprávě, kromě samotných dat (ale nemusí se zpracovávat žádné MQIIH na začátku zprávy).

Pokud vaše aplikace používá produkt *MFSMapName*, můžete místo toho použít zprávy s rozhraním MQFMT_IMS a definovat název mapování předaný transakci IMS v poli MFSMapName MQIIH.

Příjem zpráv z mostu IBM MQ - IMS

Je-li na původní zprávě, kterou odesíláte do produktu IMS, přítomna struktura MQIIH, ve zprávě s odpovědí se také nachází jedna z nich.

Chcete-li se ujistit, že je vaše odpověď správně převedena:

- Pokud máte ve své původní zprávě strukturu MQIIH, uveďte formát, který chcete pro svou zprávu odpovědi, v poli MQIIH *ReplytoFormat* původní zprávy. Tato hodnota je umístěna v poli MQIIH *Format* zprávy odpovědi. To je zvláště užitečné, pokud všechna vaše výstupní data jsou ve tvaru LLZZ < znaková data >.
- Pokud ve vaší původní zprávě nemáte strukturu MQIIH, uveďte formát, který chcete pro zprávu odpovědi jako název MFS MOD v ISRT aplikace IMS na IOVNOSE.

Vývoj aplikací JMS a Java

Produkt IBM MQ poskytuje dvě jazyková rozhraní Java : IBM MQ classes for Java Message Service a IBM MQ classes for Java.

V produktu IBM MQ jsou k dispozici dvě alternativní rozhraní API pro použití v aplikacích produktu Java :

IBM MQ classes for JMS
IBM MQ classes for Java Message Service (JMS) je poskytovatel JMS , který je dodáván s IBM MQ. Produkt Java Platform, Enterprise Edition Connector Architecture (JCA) poskytuje standardní způsob připojení aplikací spuštěných v prostředí produktu Java EE k podnikovému informačnímu systému (EIS), jako je například produkt IBM MQ nebo Db2.

IBM MQ classes for Java
IBM MQ classes for Java umožňuje používat IBM MQ v prostředí Java . IBM MQ classes for Java umožňuje aplikaci Java připojit se k IBM MQ jako klient IBM MQ nebo se připojit přímo ke správci front IBM MQ .

Poznámka:

Produkt IBM MQ classes for Java je funkčně stabilizovaný na úrovni dodávané v produktu IBM MQ 8.0. Další informace viz Stabilizace tříd produktu IBM MQ pro jazyk Java.

IBM MQ classes for Java nejsou v produktu IMSpodporovány.

IBM MQ classes for Java nejsou v produktu WebSphere Application Server Libertypodporovány. Nesmí se používat buď s funkcí systému zpráv produktu IBM MQ Liberty , ani s generickou podporou produktu JCA . Další informace naleznete v tématu Použití rozhraní Java produktu WebSphere MQ v prostředí J2EE/JEE Environments.

použitíIBM MQ classes for JMS

IBM MQ classes for Java Message Service (IBM MQ classes for JMS) je poskytovatel JMS , který je dodáván s IBM MQ. Stejně jako implementace rozhraní definovaných v balíku javax.jms poskytuje produkt IBM MQ classes for JMS dvě sady rozšíření rozhraní API produktu JMS .

Specifikace JMS definuje sadu rozhraní, které aplikace mohou používat k provádění operací systému zpráv. Poslední verze specifikace je JMS 2.0. Balík javax.jms definuje rozhraní produktu JMS a poskytovatel JMS implementuje tato rozhraní pro specifický produkt systému zpráv. IBM MQ classes for JMS je poskytovatel JMS , který implementuje rozhraní JMS pro IBM MQ.

Specifikace JMS očekává objekty ConnectionFactory a Destination Object, aby byly spravovány objekty. Administrátor vytváří a udržuje spravované objekty v centrálním úložišti a aplikace JMS načte tyto objekty pomocí produktu Java Naming Directory Interface (JNDI). Produkt IBM MQ classes for JMS podporuje použití spravovaných objektů a administrátor může k vytvoření a údržbě spravovaných objektů použít buď administrativní nástroj produktu IBM MQ JMS , nebo produkt IBM MQ Explorer .

Produkt IBM MQ classes for JMS také poskytuje dvě sady rozšíření rozhraní API produktu JMS . Hlavní zaměření těchto rozšíření se týká vytváření a konfigurace továren připojení a míst určení dynamicky za běhu, ale rozšíření také poskytují funkce, které nejsou přímo spojené se systémem zpráv, jako je funkce určování problémů.

Rozšíření produktu IBM MQ JMS
Předchozí verze produktu IBM MQ classes for JMS obsahují rozšíření, která jsou implementována v objektech, jako jsou objekty MQConnectionFactory, MQQueue a MQTopic. Tyto objekty mají

vlastnosti a metody, které jsou specifické pro produkt IBM MQ. Objekty mohou být spravovanými objekty nebo aplikace může dynamicky vytvářet objekty za běhu. Toto vydání produktu IBM MQ classes for JMS udržuje tato rozšíření, která jsou nyní známá jako rozšíření produktu IBM MQ JMS . Můžete pokračovat v používání, beze změny, jakýchkoli aplikací, které používají tato rozšíření.

Rozšíření produktu IBM JMS

Toto vydání produktu IBM MQ classes for JMS poskytuje více generické sady rozšíření rozhraní API JMS , které nejsou specifické pro systém IBM MQ jako systém zasílání zpráv. Tato rozšíření jsou známá jako rozšíření produktu IBM JMS a mají následující obecné cíle:

- Poskytovat vyšší úroveň konzistence mezi poskytovateli produktu IBM JMS
- Chcete-li usnadnit zápis do aplikace mostu mezi dvěma systémy zasílání zpráv IBM
- Usnadní portům aplikace z jednoho poskytovatele IBM JMS do jiného.

Rozšíření poskytují funkci podobnou té, která je poskytována v produktu IBM Message Service Client for C/C++ a IBM Message Service Client for .NET.

V produktu IBM MQ 8.0se produkt IBM MQ classes for JMS sestavuje s produktem Java 7. Běhové prostředí produktu Java 7 podporuje spuštění starších verzí souborů tříd.

V 9.0.0.6 IBM MQ 9.0.5 byla konečná verze Continuous Delivery pro IBM MQ 9.0. Proto se od IBM MQ 9.0.0 Fix Pack 6 aktualizuje informace Javadoc pro IBM MQ classes for JMS tak, aby odrážela chování IBM MQ classes for JMS pouze pro funkce dostupné zákazníkům Long Term Support .

Související pojmy

“Model produktu JMS” na stránce 118
Model produktu JMS definuje sadu rozhraní, která mohou aplikace produktu Java používat k provádění operací systému zpráv. IBM MQ classes for JMS, jako poskytovatel JMS definuje, jak se objekty JMS vztahují k koncepcím IBM MQ . Specifikace JMS očekává určité objekty JMS , které mají být spravovány objekty. Produkt JMS 2.0 zavádí zjednodušené rozhraní API, přičemž zachová také klasické rozhraní API z produktu JMS 1.1.

“Použití funkčnosti produktu JMS 2.0” na stránce 289
Produkt JMS 2.0 zavádí několik nových oblastí funkčnosti produktu IBM MQ classes for JMS.

Související informace

Rozhraní jazyka produktu IBM MQ Java

Proč bych měl používat produkt IBM MQ classes for JMS?

Použití produktu IBM MQ classes for JMS má řadu výhod, včetně možnosti opětovného použití existujících dovedností JMS ve vaší organizaci a aplikace jsou nezávislé na poskytovateli JMS a v základní konfiguraci IBM MQ .

IBM MQ classes for JMS je jedno ze dvou alternativních rozhraní API, které aplikace Java mohou použít pro přístup k prostředkům produktu IBM MQ . Další rozhraní API je IBM MQ classes for Java. Ačkoli existující aplikace, které používají produkt IBM MQ classes for Java , jsou i nadále plně podporovány, by nové aplikace měly používat produkt IBM MQ classes for JMS (viz “Volba rozhraní API” na stránce 71).

Souhrn výhod použití produktu IBM MQ classes for JMS

Použití produktu IBM MQ classes for JMS umožňuje opětovné použití existujících dovedností produktu JMS a zajištění nezávislosti aplikací.

- Dovednosti produktu JMS můžete znovu použít.

  IBM MQ classes for JMS je poskytovatel JMS , který implementuje rozhraní JMS pro IBM MQ jako systém zasílání zpráv. Je-li vaše organizace pro produkt IBM MQnová, ale již má zkušenosti s vývojem aplikací produktu JMS , může být snazší použít známé rozhraní API produktu JMS pro přístup k prostředkům produktu IBM MQ , nikoli k jednomu z dalších rozhraní API poskytovaných s produktem IBM MQ.
- JMS je integrální součást Java Platform, Enterprise Edition (Java EE).

JMS je přirozené rozhraní API, které má být používáno pro systém zpráv na platformě Java EE . Každý aplikační server, který je ve shodě s produktem Java EE , musí obsahovat poskytovatele JMS . Můžete použít produkt JMS pro aplikační klienty, servlety, stránky serveru Java (JSP), objekty EJB (Enterprise Java Bean) a objekty typu message-driven bean (MDB). Všimněte si zejména, že aplikace produktu Java EE používají objekty MDB k asynchronnímu zpracování zpráv a všechny zprávy jsou doručeny do objektů MDB jako zprávy produktu JMS .

- Továrny připojení a cíle lze ukládat jako spravované objekty produktu JMS v centrálním úložišti, spíše než aby byly pevně naprogramovány do aplikace.

  Administrátor může vytvářet a spravovat administrované objekty produktu JMS v centrálním úložišti a aplikace produktu IBM MQ classes for JMS mohou tyto objekty načítat pomocí produktu Java Naming Directory Interface (JNDI). JMS továrny připojení a cíle zapouzdřují specifické informace o produktu IBM MQ, jako jsou názvy správců front, názvy kanálů, volby připojení, názvy front a názvy témat. Pokud jsou továrny připojení a cíle uloženy jako spravované objekty, tyto informace nejsou do aplikace pevně naprogramovány. Toto uspořádání proto poskytuje aplikaci se stupněm nezávislosti na základní konfiguraci produktu IBM MQ .

- JMS je průmyslové standardní rozhraní API, které může poskytovat přenositelnost aplikací.

  Aplikace JMS může pomocí produktu JNDI načítat továrny připojení a cíle, které jsou uloženy jako spravované objekty, a používat pouze rozhraní, která jsou definována v balíku javax.jms k provádění operací systému zpráv. Aplikace je pak zcela nezávislá na všech poskytovatelích JMS , jako je například IBM MQ classes for JMS, a může být přenesena z jednoho poskytovatele JMS do jiného, aniž by došlo k jakékoli změně v aplikaci. Není-li produkt JNDI k dispozici v konkrétním prostředí aplikace, může aplikace IBM MQ classes for JMS použít rozšíření rozhraní API produktu JMS k dynamickému vytváření a konfiguraci továren připojení a cílů v době běhu programu. Aplikace je pak kompletně samostatná, ale je vázána k IBM MQ classes for JMS jako poskytovatel JMS .

- Aplikace mostu mohou být snazší pro zápis pomocí produktu JMS.

  Aplikace mostu je aplikace, která přijímá zprávy z jednoho systému zpráv a odesílá je do jiného systému zasílání zpráv. Psaní aplikace mostu lze komplikovat pomocí rozhraní API specifických pro produkt a formátů zpráv. Místo toho můžete napsat aplikaci mostu pomocí dvou poskytovatelů produktu JMS , jednoho pro každý systém zasílání zpráv. Aplikace pak použije pouze jedno rozhraní API, rozhraní API produktu JMS a zpracovává pouze zprávy produktu JMS .

Implementovatelná prostředí

Chcete-li zajistit integraci s aplikačním serverem produktu Java EE , vyžadují standardy produktu Java EE poskytovatele systému zpráv, aby mohli poskytovat adaptér prostředků. Na základě specifikace Java EE Connector Architecture (JCA) poskytuje produkt IBM MQ adaptér prostředků, který používá produkt JMS k poskytování funkcí systému zpráv v rámci certifikovaného prostředí Java EE .

I když bylo možné použít IBM MQ classes for Java uvnitř Java EE, není toto rozhraní konstruováno nebo optimalizováno pro tento účel. Viz technická poznámka IBM Použití rozhraní Java WebSphere MQ Java v prostředí J2EE/JEE Environments pro podrobné informace o aspektech IBM MQ classes for Java v rámci produktu Java EE.

Mimo prostředí produktu Java EE jsou k dispozici soubory OSGi a JAR, takže je pro vás snazší získat pouze IBM MQ classes for JMS. Tyto soubory JAR jsou nyní lépe implementovatelné buď samostatné, nebo v rámci rámců správy softwaru, jako je nástroj Maven. Další informace naleznete v technické poznámce IBM Získání tříd produktu WebSphere MQ pro službu JMS.

Volba rozhraní API

Nové aplikace by měly používat spíše IBM MQ classes for JMS než IBM MQ classes for Java.

Produkt IBM MQ classes for JMS poskytuje přístup k funkcím systému zpráv typu IBM MQmezi dvěma body i pro publikování/odběr. Kromě odesílání zpráv produktu JMS , které poskytují podporu pro standardní model systému zpráv produktu JMS , mohou aplikace také odesílat a přijímat zprávy bez dalších záhlaví a mohou spolupracovat s dalšími aplikacemi produktu IBM MQ , například s aplikacemi rozhraní C MQI. Úplný ovládací prvek informačního obsahu zpráv MQMD a MQ je k dispozici. K dispozici

jsou také další funkce produktu IBM MQ , jako např. streaming zpráv, asynchronní vkládání zpráv a zpráv sestav. Using the supplied PCF helper classes, IBM MQ PCF administration messages can be sent and received through the JMS API and can be used to administer queue managers.

Funkce, které byly nedávno přidány do produktu IBM MQ, jako např. asynchronní spotřeba a automatické opětovné připojení, nejsou v produktu IBM MQ classes for Javak dispozici, ale jsou k dispozici v produktu IBM MQ classes for JMS. Existující aplikace, které používají produkt IBM MQ classes for Java , jsou nadále plně podporovány.

Potřebujete-li přístup k funkcím produktu IBM MQ , které nejsou k dispozici prostřednictvím produktu IBM MQ classes for JMS, můžete zvýšit požadavek na vylepšení (RFE). Produkt IBM pak může informovat o tom, zda je implementace možná v implementaci produktu IBM MQ classes for JMS , nebo zda existuje nejlepší postup, který lze sledovat. Pro další funkce systému zpráv, protože produkt IBM je přispěvatelem na otevřený standard, lze tyto funkce zobrazit jako část procesu JCP.

Související informace

Proces odeslání IBM RFE
Proces přezkoumání specifikace Java JMS
Použití rozhraní Java produktu WebSphere MQ v prostředí J2EE/JEE Environments
Získání tříd produktu WebSphere MQ pro službu JMS
Použití JMS k odeslání zpráv PCF
Trasování aplikací produktu IBM MQ classes for JMS
Odstraňování problémů s Java a JMS

Nezbytné předpoklady pro IBM MQ classes for JMS

Toto téma uvádí informace, které musíte znát před použitím produktu IBM MQ classes for JMS. Chcete-li vyvíjet a spouštět aplikace IBM MQ classes for JMS , potřebujete určité softwarové komponenty jako nezbytné předpoklady.

Chcete-li získat informace o předpokladech pro IBM MQ classes for JMS, prohlédněte si téma Systémové požadavky pro IBM MQ.

Chcete-li vyvíjet aplikace IBM MQ classes for JMS , potřebujete sadu SDK (Software Development Kit) produktu Java SE . Podrobnosti o JDK podporovaných vaším operačním systémem viz Systémové požadavky pro IBM MQ.

Chcete-li spustit aplikace produktu IBM MQ classes for JMS , potřebujete tyto softwarové komponenty:

- Správce front produktu IBM MQ .
- Java runtime environment (JRE) pro každý systém, na kterém spouštíte aplikace.
- IBM i Pro IBM i, Qshell, což je volba 30 operačního systému.
- z/OS Pro produkt z/OS, UNIX and Linux System Services (USS).

V 9.0.0 Poskytovatel JSSE IBM zahrnuje poskytovatele certifikovaného šifrování FIPS, takže může být programově konfigurován pro standard FIPS 140-2 připravený k okamžitému použití. Proto může být kompatibilita FIPS 140-2 podporována přímo z IBM MQ classes for Java a IBM MQ classes for JMS.

V 9.0.0 Poskytovatel JSSE Oraclemůže mít poskytovatele certifikovaného šifrování FIPS, který je v něm nakonfigurován, ale není připraven k okamžitému použití a není k dispozici pro programovou konfiguraci. Proto v tomto případě IBM MQ classes for Java a IBM MQ classes for JMS nemohou povolit shodu FIPS 140-2 přímo. Můžete být schopni ručně povolit takové dodržování předpisů (opět si prohlédněte diskusi na Vyhovující režim FIPS 140 pro SunJSSE u některých ukazatelů), ale IBM v současné době v tomto případě nemůže poskytnout vodítko.

Adresy Internet Protocol verze 6 (IPv6) můžete použít ve svých aplikacích IBM MQ classes for JMS , pokud jsou adresy IPv6 podporovány vaším virtuálním počítačem Java (JVM) a implementací TCP/IP ve vašem operačním systému. Administrativní nástroj produktu IBM MQ JMS (viz téma Konfigurace objektů produktu JMS pomocí nástroje pro administraci) také přijímá adresy IPv6 .

Administrativní nástroj produktu IBM MQ JMS a produkt IBM MQ Explorer používají produkt Java Naming Directory Interface (JNDI) pro přístup k adresářové službě, ve které jsou uloženy spravované objekty. Aplikace produktu IBM MQ classes for JMS mohou také pomocí produktu JNDI načítat spravované objekty z adresářové služby. Poskytovatel služeb je kód, který poskytuje přístup k adresářové službě mapováním JNDI volání do adresářové služby. Poskytovatel služby systému souborů v souborech `fscontext.jar` a `providerutil.jar` je dodáván s produktem IBM MQ classes for JMS. Poskytovatel služeb systému souborů poskytuje přístup k adresářové službě založené na lokálním systému souborů.

Hodláte-li používat adresářovou službu založenou na serveru LDAP, musíte nainstalovat a nakonfigurovat server LDAP, nebo mít přístup k existujícímu serveru LDAP. Zejména musíte nakonfigurovat server LDAP pro ukládání objektů Java . Informace o tom, jak instalovat a konfigurovat server LDAP, najdete v dokumentaci dodané se serverem.

Instalace a konfigurace produktu IBM MQ classes for JMS

Tato sekce popisuje adresáře a soubory, které jsou vytvořeny při instalaci produktu IBM MQ classes for JMS a uvádí, jak nakonfigurovat produkt IBM MQ classes for JMS po instalaci.

Související pojmy

“Co je nainstalováno pro IBM MQ classes for JMS” na stránce 74
Při instalaci produktu IBM MQ classes for JMSse vytvoří mnoho souborů a adresářů. V systému Windowsse při instalaci provádí některá konfigurace automaticky nastavováním proměnných prostředí. Na jiných platformách a v některých prostředích produktu Windows musíte nastavit proměnné prostředí před spuštěním aplikací produktu IBM MQ classes for JMS .

“Spuštění aplikací produktu IBM MQ classes for JMS v rámci struktury Java Security Manager” na stránce 89
Produkt IBM MQ classes for JMS může být spuštěn se zapnutým správcem zabezpečení produktu Java . Chcete-li úspěšně spustit aplikace se zapnutým serverem Java Security Manager , musíte nakonfigurovat prostředí Java virtual machine (JVM) s vhodným konfiguračním souborem zásad.

“Použití adaptéru prostředků produktu IBM MQ” na stránce 400
Adaptér prostředků umožňuje aplikacím běžícím na aplikačním serveru přistupovat k prostředkům produktu IBM MQ . Podporuje příchozí a odchozí komunikaci.

“Postup po instalaci pro aplikace produktu IBM MQ classes for JMS” na stránce 91
Toto téma informuje o tom, jaké oprávnění aplikace IBM MQ classes for JMS potřebují, aby mohli přistupovat k prostředkům správce front. Také uvádí režimy připojení a popisuje, jak nakonfigurovat správce front, aby se aplikace mohly připojit v režimu klienta.

“Dvoubodové IVT pro IBM MQ classes for JMS” na stránce 94
Program testu IVT (point-to-point installation test) se dodává s IBM MQ classes for JMS. Program se připojuje ke správci front v obou vazbách nebo v režimu klienta, odesílá zprávu do fronty s názvem SYSTEM.DEFAULT.LOCAL.QUEUE, a poté přijme zprávu z fronty. Program může vytvořit a konfigurovat všechny objekty, které vyžaduje dynamicky za běhu, nebo může použít rozhraní JNDI k načtení spravovaných objektů z adresářové služby.

“Publikování/odběr IVT pro IBM MQ classes for JMS” na stránce 98
Program IVT (publish/subscribe installation verification test) je dodáván s produktem IBM MQ classes for JMS. Program se připojí ke správci front buď v rámci vazeb, nebo v režimu klienta, přihlásí se k odběru tématu, publikuje zprávu na daném tématu a poté přijme zprávu, kterou právě publikoval. Program může vytvořit a konfigurovat všechny objekty, které vyžaduje dynamicky za běhu, nebo může použít rozhraní JNDI k načtení spravovaných objektů z adresářové služby.

“Konfigurace adaptéru prostředků pro odchozí komunikaci” na stránce 431
Chcete-li nakonfigurovat odchozí komunikaci, definujte vlastnosti objektu ConnectionFactory a spravovaného cílového objektu.

“Podpora pro OSGi” na stránce 106
OSGi poskytuje rámec, který podporuje implementaci aplikací jako balíky. Devět svazků balíků OSGi je dodáváno jako součást produktu IBM MQ classes for JMS.

Související úlohy

“Ověření instalace adaptéru prostředků” na stránce 449

Program IVT (installation verification test) pro adaptér prostředků produktu IBM MQ je dodáván jako soubor EAR. Chcete-li použít tento program, musíte jej implementovat a definovat některé objekty jako prostředky produktu JCA .

Související odkazy

“Skripty poskytnuté s IBM MQ classes for JMS” na stránce 105
Je k dispozici řada skriptů, které pomáhají s běžnými úlohami, které je třeba provést při používání produktu IBM MQ classes for JMS.

Související informace

Odstraňování problémů s IBM MQ classes for JMS
Určování problémů pro adaptér prostředků produktu IBM MQ

Co je nainstalováno pro IBM MQ classes for JMS

Při instalaci produktu IBM MQ classes for JMSse vytvoří mnoho souborů a adresářů. V systému Windowsse při instalaci provádí některá konfigurace automaticky nastavováním proměnných prostředí. Na jiných platformách a v některých prostředích produktu Windows musíte nastavit proměnné prostředí před spuštěním aplikací produktu IBM MQ classes for JMS .

Pro většinu operačních systémů je produkt IBM MQ classes for JMS instalován jako volitelná komponenta při instalaci produktu IBM MQ.

Další informace o instalaci produktu IBM MQnaleznete v následujících tématech:

Multi Instalace produktu IBM MQ
z/OS Instalace produktu IBM MQ for z/OS

Důležité:

- Kromě přemístitelných souborů JAR popsaných v tomto tématu není podporováno kopírování souborů JAR produktu IBM MQ classes for JMS nebo nativních knihoven do jiných počítačů nebo do jiného umístění na počítači, na kterém byl nainstalován produkt IBM MQ classes for JMS .
- Kromě toho není podporováno, včetně souboru `com.ibm.mq.allclient.jar` nebo IBM MQ classes for JMS, v archivech aplikace (jako jsou například archivy podnikových aplikací nebo soubory EAR).

Měli byste se proto vyhnout vytváření balíků souborů JAR produktu IBM MQ ve vašich aplikacích (soubory EAR na systému WebSphere Application Server), jinak se můžete setkat s neočekávanými problémy spojenými se spuštěnou úrovní back-level, bez opravy kódu.

Instalační adresáře

Tabulka 5 na stránce 74 zobrazuje, kde jsou soubory IBM MQ classes for JMS nainstalovány na každé platformě.

Tabulka 5. Instalační adresáře produktu IBM MQ classes for JMS

Platforma	**Adresář**
AIX UNIX and Linux	*MQ_INSTALLATION_PATH*`/java`
Windows Windows	*MQ_INSTALLATION_PATH*`\java`
IBM i IBM i	`/QIBM/ProdData/mqm/java`
z/OS z/OS	*MQ_INSTALLATION_PATH*`/mqm/V9R0M0/java`
	MQ_INSTALLATION_PATH`/opt/mqm/java`

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Instalační adresář obsahuje následující obsah:

- Soubory JAR IBM MQ classes for JMS , které se nacházejí v adresáři *MQ_INSTALLATION_PATH*\java\lib .
- Nativní knihovny produktu IBM MQ , které jsou používány aplikacemi, které používají nativní rozhraní produktu Java .

  32bitové nativní knihovny jsou instalovány do adresáře *MQ_INSTALLATION_PATH*\java\lib a 64bitové nativní knihovny lze najít v adresáři *MQ_INSTALLATION_PATH*\java\lib64 .

  Další informace o nativních knihovnách produktu IBM MQ viz "Konfigurace knihoven JNI (Java Native Interface)" na stránce 79.
- Další skripty, které jsou popsány v tématu "Skripty poskytnuté s IBM MQ classes for JMS" na stránce 105. Tyto skripty se nacházejí v adresáři *MQ_INSTALLATION_PATH*\java\bin .
- Specifikace rozhraní API produktu IBM MQ classes for JMS . Nástroj Javadoc se používá ke generování stránek HTML, které obsahují specifikace rozhraní API.

  Stránky HTML se nacházejí v adresáři *MQ_INSTALLATION_PATH*\java\doc\WMQJMSClasses :

  – ULW V systému UNIX, Linux, and Windowstento podadresář obsahuje jednotlivé stránky HTML.

  – IBM i V systému IBM ise stránky HTML nacházejí v souboru s názvem wmqjms_javadoc.jar.

  – z/OS V systému z/OSse stránky HTML nacházejí v souboru s názvem wmqjms_javadoc.jar.
- Podpora pro OGSi. Balíky OSGi jsou nainstalovány v adresáři java\lib\OSGi a popsány v "Podpora pro OSGi" na stránce 106.
- Adaptér prostředků produktu IBM MQ , který lze implementovat do libovolného aplikačního serveru vyhovujícího produktu Java Platform, Enterprise Edition 7 (Java EE 7).

  Adaptér prostředků produktu IBM MQ se nachází v adresáři *MQ_INSTALLATION_PATH*\java\lib\jca ; další informace viz "Použití adaptéru prostředků produktu IBM MQ" na stránce 400 .
- Windows V systému Windowsse symboly, které lze použít pro ladění, instalují do adresáře *MQ_INSTALLATION_PATH*\java\lib\symbols .

Instalační adresář také obsahuje některé soubory, které patří do jiných komponent produktu IBM MQ :

- Přenos protokolu IBM MQ pro SOAP, který poskytuje přenos JMS pro SOAP, je nainstalován do adresáře *MQ_INSTALLATION_PATH*\java\lib\soap . Další informace o transportu IBM MQ pro protokol SOAP naleznete v tématu "Vyvíjení webových služeb s přenosem produktu IBM MQ pro SOAP" na stránce 1256.

  V produktu IBM MQ 9.0je přenos IBM MQ pro protokol SOAP zamítnutý.

V 9.0.0.3 V 9.0.5 Soubor JSON4J.jar a balík com.ibm.msg.client.mqlight nejsou vyžadovány IBM MQ classes for Java a IBM MQ classes for JMS. V produktu IBM MQ 9.0.0 Fix Pack 3 a IBM MQ 9.0.5jsou proto provedeny následující změny souboru com.ibm.mq.allclient.jar :

- Odkaz na soubor JSON4J.jar se odstraní ze souboru cesty ke třídě v souboru typu manifest pro soubor com.ibm.mq.allclient.jar .
- Balík com.ibm.msg.client.mqlight již není zahrnut do souboru com.ibm.mq.allclient.jar .

Ukázkové aplikace

Některé ukázkové aplikace jsou dodávány s produktem IBM MQ classes for JMS. Tabulka 6 na stránce 76 zobrazuje, kde jsou ukázkové aplikace nainstalovány na každé platformě.

Tabulka 6. Adresáře ukázek	
Platforma	**Adresář**
AIX UNIX and Linux	*MQ_INSTALLATION_PATH*/samp/jms
Windows Windows	*MQ_INSTALLATION_PATH*\tools\jms
IBM i IBM i	/QIBM/ProdData/mqm/java/samples/jms
z/OS z/OS	*MQ_INSTALLATION_PATH*/mqm/V9R0M0/java/samples/jms
	MQ_INSTALLATION_PATH/opt/mqm/samp/jms

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Po instalaci možná budete muset provést některé konfigurační úlohy pro kompilaci a spuštění aplikací.

"Nastavení proměnných prostředí" na stránce 77 popisuje cestu ke třídě, která je nezbytná ke spouštění jednoduchých aplikací produktu IBM MQ classes for JMS . Toto téma také popisuje další soubory JAR, které je třeba odkazovat ve zvláštních situacích, a proměnné prostředí, které musíte nastavit ke spuštění skriptů poskytnutých s produktem IBM MQ classes for JMS.

Chcete-li ovládat vlastnosti, jako je například trasování a protokolování aplikace, musíte poskytnout soubor vlastností konfigurace. Soubor vlastností konfigurace produktu IBM MQ classes for JMS je popsán v tématu "Konfigurační soubor IBM MQ classes for JMS" na stránce 81.

Přeložitelné soubory JAR

V rámci podniku lze přesunout následující soubory na systémy, které potřebují spustit produkt IBM MQ classes for JMS:

- `-com.ibm.mq.allclient.jar`
- `-com.ibm.mq.traceControl.jar`
- `-jms.jar`
- `-fscontext.jar`
- `-providerutil.jar`
- Poskytovatel zabezpečení Bouncy Castle Security a soubory JAR podpory CMS

Soubory `fscontext.jar` a `providerutil.jar` jsou vyžadovány, pokud vaše aplikace provádí vyhledávání JNDI pomocí kontextu systému souborů.

Je požadován poskytovatel zabezpečení Bouncy Castle Security a soubory JAR podpory CMS. Další informace naleznete v tématu Podpora pro jiná prostředí nežIBM JRE. Požadují se následující soubory JAR:

- `bcpkix-jdk15on.jar`
- `bcprov-jdk15on.jar`
- V 9.0.0.12 `bcutil-jdk15on.jar`

Soubor `com.ibm.mq.allclient.jar` obsahuje IBM MQ classes for JMS, IBM MQ classes for Javaa třídy PCF a Headers. Přesunete-li tento soubor do nového umístění, ujistěte se, že jste provedli kroky k zachování tohoto nového umístění s novými opravami Fix Pack produktu IBM MQ . Také se ujistěte, že je použití tohoto souboru známo podpoře produktu IBM , pokud máte prozatímní opravu.

Chcete-li určit verzi souboru `com.ibm.mq.allclient.jar`, použijte následující příkaz:

```
java -jar com.ibm.mq.allclient.jar
```

Následující příklad zobrazuje ukázkový výstup z tohoto příkazu:

```
C:\Program Files\IBM\MQ_1\java\lib>java -jar com.ibm.mq.allclient.jar
Name:       Java Message Service Client
Version:    9.0.0.0
Level:      p000-L140428.1
Build Type: Production
Location:   file:/C: /Program Files/IBM/MQ_1/java/lib/com.ibm.mq.allclient.jar

Name:       WebSphere MQ classes for Java Message Service
Version:    9.0.0.0
Level:      p000-L140428.1
Build Type: Production
Location:   file:/C: /Program Files/IBM/MQ_1/java/lib/com.ibm.mq.allclient.jar

Name:       WebSphere MQ JMS Provider
Version:    9.0.0.0
Level:      p000-L140428.1 mqjbnd=p000-L140428.1
Build Type: Production
Location:   file:/C: /Program Files/IBM/MQ_1/java/lib/com.ibm.mq.allclient.jar

Name:       Common Services for Java Platform, Standard Edition
Version:    9.0.0.0
Level:      p000-L140428.1
Build Type: Production
Location:   file:/C: /Program Files/IBM/MQ_1/java/lib/com.ibm.mq.allclient.jar
```

Soubor `com.ibm.mq.traceControl.jar` se používá k dynamickému řízení trasování pro aplikace produktu IBM MQ classes for JMS . Další informace viz Řízení trasování ve spuštěném procesu pomocí tříd produktu IBM MQ pro třídy Java a IBM MQ pro platformu JMS.

Související informace

Problémy při implementaci adaptéru prostředků

Nastavení proměnných prostředí

Než budete moci kompilovat a spustit aplikace produktu IBM MQ classes for JMS , musí nastavení proměnné prostředí CLASSPATH obsahovat soubor archivu IBM MQ classes for JMS Java (JAR).
V závislosti na vašich požadavcích budete možná muset do své cesty ke třídě přidat další soubory JAR.
Chcete-li spustit skripty poskytnuté s produktem IBM MQ classes for JMS, musí být nastaveny další proměnné prostředí.

Informace o této úloze

Důležité: Nastavení volby Java *-Xbootclasspath*tak, aby zahrnovalo IBM MQ classes for JMS, není podporováno.

Chcete-li zkompilovat a spustit aplikace produktu IBM MQ classes for JMS , použijte nastavení CLASSPATH pro vaši platformu, jak je zobrazeno v tématu Tabulka 7 na stránce 77. Nastavení obsahuje adresář ukázek, takže je možné kompilovat a spustit ukázkové aplikace produktu IBM MQ classes for JMS . Jinou možností je zadání cesty ke třídám v příkazu **java** namísto použití proměnné prostředí.

Tabulka 7. Nastavení CLASSPATH pro kompilaci a spuštění aplikací produktu IBM MQ classes for JMS , včetně ukázkových aplikací

Platforma	Nastavení CLASSPATH
AIX	CLASSPATH= *MQ_INSTALLATION_PATH*/java/lib/com.ibm.mqjms.jar: *MQ_INSTALLATION_PATH*/samp/jms/samples:
	CLASSPATH= *MQ_INSTALLATION_PATH*/java/lib/com.ibm.mqjms.jar: *MQ_INSTALLATION_PATH*/samp/jms/samples:

Tabulka 7. Nastavení CLASSPATH pro kompilaci a spuštění aplikací produktu IBM MQ classes for JMS , včetně ukázkových aplikací (pokračování)	
Platforma	**Nastavení CLASSPATH**
Solaris Linux HP-UX HP-UX, Linuxa Solaris	CLASSPATH= *MQ_INSTALLATION_PATH*/java/lib/com.ibm.mqjms.jar: *MQ_INSTALLATION_PATH*/samp/jms/samples:
IBM i IBM i	CLASSPATH=/QIBM/ProdData/mqm/java/lib/com.ibm.mqjms.jar: /QIBM/ProdData/mqm/java/samples/jms/samples:
Windows Windows	CLASSPATH= *MQ_INSTALLATION_PATH*\java\lib\com.ibm.mqjms.jar; *MQ_INSTALLATION_PATH*\tools\jms\samples;
z/OS z/OS	CLASSPATH= *MQ_INSTALLATION_PATH*/mqm/V9R0M0/java/lib/ com.ibm.mqjms.jar: *MQ_INSTALLATION_PATH*/mqm/V9R0M0/java/samples/jms/samples:

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Soubor typu manifest souboru JAR `com.ibm.mqjms.jar` obsahuje odkazy na většinu dalších souborů JAR vyžadovaných aplikacemi produktu IBM MQ classes for JMS , a proto nemusíte přidávat tyto soubory JAR do cesty ke třídám. Tyto soubory JAR zahrnují ty, které jsou vyžadovány aplikacemi, které používají produkt Java Naming Directory Interface (JNDI) k načtení spravovaných objektů z adresářové služby a aplikací, které používají rozhraní JTA (Java Transaction API).

Do cesty ke třídě však musíte zahrnout další soubory JAR za následujících okolností:

- Pokud používáte třídy ukončení kanálu, které implementují rozhraní uživatelské procedury kanálu definovaná v balíku produktu `com.ibm.mq` místo těch, které jsou definovány v balíku produktu `com.ibm.mq.exits` , musíte přidat soubor JAR IBM MQ classes for Java `com.ibm.mq.jar`do cesty ke třídě.
- Pokud vaše aplikace používá JNDI k načtení spravovaných objektů z adresářové služby, musíte do své cesty ke třídě přidat také následující soubory JAR:
  - `fscontext.jar`
  - `providerutil.jar`
- Pokud vaše aplikace používá JTA, musíte také přidat `jta.jar` do své cesty ke třídě.

Poznámka: Tyto další soubory JAR jsou nezbytné pouze pro kompilaci vašich aplikací, nikoli pro jejich spuštění.

Skripty poskytnuté s produktem IBM MQ classes for JMS používají následující proměnné prostředí:

MQ_JAVA_DATA_PATH
Tato proměnná prostředí určuje adresář pro výstup protokolu a trasování.

INSTALAČNÍ_CESTA MQ_JAVA_INSTALL_PATH
Tato proměnná prostředí určuje adresář, kde je nainstalován produkt IBM MQ classes for JMS .

KOŘEN ROZHRANÍ MQ_JAVA_LIB_PATH
Tato proměnná prostředí určuje adresář, kde jsou uloženy knihovny produktu IBM MQ classes for JMS , jak je zobrazeno v Tabulka 8 na stránce 80.

Procedura

- Windows
  V systému Windowsspusťte po instalaci produktu IBM MQpříkaz **setmqenv**.
  Pokud tento příkaz nespustíte jako první, může se při zadávání příkazu **dspmqver** zobrazit následující chybová zpráva:

  V 9.0.2 `Prostředí Java AMQ8351: IBM MQ nebylo nakonfigurováno Správně, nebo funkce prostředí IBM MQ JRE nebyla nainstalována.`

  Poznámka: V 9.0.2 Tato zpráva se má očekávat, pokud jste neinstalovali prostředí JRE (Java Runtime Environment) produktu IBM MQ .
- Na jakékoli jiné platformě nastavte proměnné prostředí sami:
  - Linux UNIX Chcete-li nastavit proměnné prostředí při použití 32bitového prostředí JVM v systémech UNIX, nebo Linux , můžete použít skript setjmsenv.
  - Linux UNIX Chcete-li nastavit proměnné prostředí při použití 64bitového prostředí JVM v systému UNIX nebo Linux , můžete použít skript setjmsenv64. Tyto skripty jsou uloženy v adresáři *MQ_INSTALLATION_PATH*`/java/bin` , kde *MQ_INSTALLATION_PATH* představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

  Skript setjmsenv nebo setjmsenv64 můžete použít různými způsoby: Můžete jej použít jako základ pro nastavení požadovaných proměnných prostředí, jak je zobrazeno v tabulce, nebo je můžete přidat do produktu `.profile` pomocí textového editoru. Máte-li netypické nastavení, upravte obsah skriptu podle potřeby. Případně můžete spustit skript v každé relaci, ze které se spustí spouštěcí skripty produktu JMS . Vyberete-li tuto volbu, musíte spustit skript v každém okně shellu, který spustíte, během procesu verifikace JMS zadáním `. ./setjmsenv` nebo `. ./setjmsenv64`

  IBM i V systému IBM ije třeba nastavit proměnnou prostředí QIBM_MULTI_THREADED na hodnotu Y. Poté můžete spustit více aplikací s podporou podprocesů stejným způsobem, jakým spouštíte jednotlivé aplikace s podporou podprocesů. Další informace naleznete v tématu Nastavení produktu IBM MQ pomocí jazyka Java a platformy JMS .

Konfigurace knihoven JNI (Java Native Interface)
Aplikace produktu IBM MQ classes for JMS , které se buď připojují ke správci front s použitím přenosu vazeb nebo které se připojují ke správci front s použitím přenosu klienta a používají ukončovací programy kanálu napsané v jiných jazycích než Java, je třeba spustit v prostředí, které umožňuje přístup ke knihovnám rozhraní JNI (Java Native Interface).

Informace o této úloze

Chcete-li nastavit toto prostředí, je třeba nakonfigurovat cestu ke knihovně prostředí tak, aby prostředí Java virtual machine (JVM) mohlo načíst knihovnu mqjbnd před spuštěním aplikace IBM MQ classes for JMS .

Produkt IBM MQ poskytuje dvě knihovny rozhraní JNI (Java Native Interface):

mqjbnd
Tato knihovna je používána aplikacemi, které se připojují ke správci front pomocí přenosu vazeb. Poskytuje rozhraní mezi produktem IBM MQ classes for JMS a správcem front. Knihovnu mqjbnd nainstalované s produktem IBM MQ 9.0 lze použít pro připojení k libovolnému správci front produktu IBM MQ 9.0 (nebo starším).

mqjexitstub02
Knihovna mqjexitstub02 je načtena produktem IBM MQ classes for JMS při připojení aplikace ke správci front pomocí přenosu klienta a používá ukončovací program kanálu, který je napsán v jiném jazyce než Java.

Na určitých platformách instaluje produkt IBM MQ 32bitové a 64bitové verze těchto knihoven JNI. Umístění knihoven pro každou platformu je zobrazeno v Tabulka 1.

Tabulka 8. Umístění knihoven produktu IBM MQ classes for JMS pro každou platformu	
Platforma	**Adresář obsahující knihovny IBM MQ classes for JMS**
AIX AIX HP-UX Linux Linux (platformy POWER, x86-64 a zSeries s390x) Solaris Solaris (platformyx86-64 a SPARC)	*MQ_INSTALLATION_PATH*/java/lib (32bitové knihovny) *MQ_INSTALLATION_PATH*/java/lib64 (64bitové knihovny)
Windows Windows	*MQ_INSTALLATION_PATH*\java\lib (32bitové knihovny) *MQ_INSTALLATION_PATH*\java\lib64 (64-bitové knihovny)
z/OS z/OS	*MQ_INSTALLATION_PATH*/mqm/V8R0M0/java/lib (31bitové a 64bitové knihovny)

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Poznámka: z/OS V systému z/OSmůžete použít buď 31bitový, nebo 64-bitový Java virtual machine (JVM). Nemusíte určovat, které knihovny JNI se mají použít; IBM MQ classes for JMS může určit, které knihovny JNI se mají načíst.

Postup

1. Zkonfigurujte vlastnost **`java.library.path`** prostředí JVM, kterou lze provést dvěma způsoby:
   - Uvedením argumentu JVM, jak je zobrazeno v následujícím příkladu:

     ```
     -Djava.library.path=path_to_library_directory
     ```

     Linux Například pro 64bitové prostředí JVM v systému Linux pro instalaci výchozího umístění zadejte:

     ```
     -Djava.library.path=/opt/mqm/java/lib64
     ```

   - Konfigurací prostředí shellu tak, aby prostředí JVM nastavila své vlastní `java.library.path`. Tato cesta se liší podle platformy a umístění, do kterého jste nainstalovali produkt IBM MQ. Například pro 64bitové prostředí JVM a pro výchozí umístění instalace produktu IBM MQ můžete použít následující nastavení:

     AIX
     ```
     export LIBPATH=/usr/mqm/java/lib64:$LIBPATH
     ```

     Solaris Linux HP-UX
     ```
     export LD_LIBRARY_PATH=/opt/mqm/java/
     lib64:$LD_LIBRARY_PATH
     ```

     Windows
     ```
     set PATH=C:\Program Files\IBM\MQ\java\lib64;%PATH%
     ```

   Příklad zásobníku výjimek, který vidíte v situaci, kdy prostředí nebylo správně nakonfigurováno, je následující:

   ```
   Příčina: com.ibm.mq.jmqi.local.LocalMQ$4: CC=2;RC=2495;
   AMQ8598: Nezdařilo se načíst nativní knihovnu JNI produktu WebSphere MQ : 'mqjbnd'.
   ```

```
    na adrese com.ibm.mq.jmqi.local.LocalMQ.loadLib(LocalMQ.java:1268)
    na adrese com.ibm.mq.jmqi.local.LocalMQ$1.run(LocalMQ.java:309)
    v java.security.AccessController.doPrivileged(AccessController.java:400)
    na adrese com.ibm.mq.jmqi.local.LocalMQ.initialise_inner(LocalMQ.java:259)
    na adrese com.ibm.mq.jmqi.local.LocalMQ.initialise(LocalMQ.java:221)
    na adrese com.ibm.mq.jmqi.local.LocalMQ. < init> (LocalMQ.java:1350)
    na adrese com.ibm.mq.jmqi.local.LocalServer. < init> (LocalServer.java:230)
    at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native metoda)
    at
sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:86)

    at
sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.jav
a:58).
    at java.lang.reflect.Constructor.newInstance(Constructor.java:542)
    na adrese com.ibm.mq.jmqi.JmqiEnvironment.getInstance(JmqiEnvironment.java:706)
    na adrese com.ibm.mq.jmqi.JmqiEnvironment.getMQI(JmqiEnvironment.java:640)
    na adrese
com.ibm.msg.client.wmq.factories.WMQConnectionFactory.createV7ProviderConnection(WMQConnectionF
actory.java:8437)
    ... 7 dalších
Příčina: java.lang.UnsatisfiedLinkError: mqjbnd (není nalezeno v java.library.path)
    at java.lang.ClassLoader.loadLibraryWithPath(ClassLoader.java:1235)
    na java.lang.ClassLoader.loadLibraryWithClassLoader(ClassLoader.java:1205)
    na java.lang.System.loadLibrary(System.java:534)
    na adrese com.ibm.mq.jmqi.local.LocalMQ.loadLib(LocalMQ.java:1240)
    ... 20 dalších
```

2. Po nastavení 32bitového nebo 64bitového prostředí spusťte aplikaci produktu IBM MQ classes for JMS pomocí následujícího příkazu:

   ```
   java application-name
   ```

   kde *název-aplikace* je název aplikace produktu IBM MQ classes for JMS , která má být spuštěna.

   IBM MQ classes for JMS vyvolá výjimku obsahující kód příčiny IBM MQ 2495 (MQRC_MODULE_NOT_FOUND), pokud:

   - Aplikace IBM MQ classes for JMS se spouští ve 32bitovém produktu Java runtime environmenta pro produkt IBM MQ classes for JMSbylo nastaveno 64bitové prostředí, protože 32bitový systém Java runtime environment nemůže načíst 64bitovou knihovnu Java nativní knihovny.
   - Aplikace IBM MQ classes for JMS je spuštěna v 64bitovém prostředí Java runtime environmenta pro prostor IBM MQ classes for JMSbylo nastaveno 32bitové prostředí, protože 64bitový produkt Java runtime environment nemůže načíst 32bitovou nativní knihovnu Java .

Konfigurační soubor IBM MQ classes for JMS
Konfigurační soubor IBM MQ classes for JMS určuje vlastnosti, které se používají ke konfiguraci produktu IBM MQ classes for JMS.

Poznámka: Vlastnosti definované v konfiguračním souboru mohou být také nastaveny jako systémové vlastnosti JVM. Je-li vlastnost nastavena v konfiguračním souboru i jako systémová vlastnost, má přednost vlastnost systému. Proto, je-li to nutné, můžete přepsat jakoukoli vlastnost v konfiguračním souboru tak, že ji uvedete jako systémovou vlastnost v příkazu **java** .

Formát konfiguračního souboru IBM MQ classes for JMS je standardní soubor vlastností Java . Ukázkový konfigurační soubor s názvem `jms.config` se dodává v podadresáři `bin` instalačního adresáře produktu IBM MQ classes for JMS . Tento soubor dokumentuje všechny podporované vlastnosti a jejich výchozí hodnoty.

Můžete zvolit název a umístění konfiguračního souboru IBM MQ classes for JMS . Když spustíte aplikaci, použijte příkaz **java** s následujícím formátem:

```
java -Dcom.ibm.msg.client.config.location= config_file_url application_name
```

V příkazu *config_file_url* je adresa URL (Uniform Resource Locator), která určuje název a umístění konfiguračního souboru IBM MQ classes for JMS . Podporovány jsou adresy URL následujících typů: http, file, ftp, and jar.

Zde je příklad příkazu **java** :

```
java -Dcom.ibm.msg.client.config.location=file:/D:/mydir/myjms.config MyAppClass
```

Tento příkaz identifikuje konfigurační soubor IBM MQ classes for JMS jako soubor `D:\mydir\mjms.config` v lokálním systému Windows .

Když se spustí aplikace, produkt IBM MQ classes for JMS načte obsah konfiguračního souboru a uloží zadané vlastnosti do interního úložiště vlastností. Pokud příkaz **java** neidentifikuje konfigurační soubor, nebo pokud nelze konfigurační soubor nalézt, použije produkt IBM MQ classes for JMS výchozí hodnoty pro všechny vlastnosti.

Konfigurační soubor produktu IBM MQ classes for JMS lze použít s libovolným z podporovaných přenosů mezi aplikací a správcem front nebo zprostředkovatelem.

Potlačení vlastností zadaných v konfiguračním souboru IBM MQ MQI client

Konfigurační soubor IBM MQ MQI client může také určovat vlastnosti, které se používají ke konfiguraci produktu IBM MQ classes for JMS. Vlastnosti zadané v konfiguračním souboru IBM MQ MQI client se však používají pouze v případě, že se aplikace připojuje ke správci front v režimu klienta.

Je-li to nutné, můžete přepsat jakýkoli atribut v konfiguračním souboru IBM MQ MQI client tak, že jej uvedete jako vlastnost do konfiguračního souboru IBM MQ classes for JMS . Chcete-li přepsat atribut v konfiguračním souboru IBM MQ MQI client , použijte záznam s následujícím formátem v konfiguračním souboru IBM MQ classes for JMS :

```
com.ibm.mq.cfg. stanza. propName = propValue
```

Proměnné v položce mají následující význam:

sekce
Název stanzy v konfiguračním souboru IBM MQ MQI client , který obsahuje atribut

propName
Název atributu, jak je uveden v konfiguračním souboru IBM MQ MQI client

propValue
Hodnota vlastnosti, která přepíše hodnotu atributu uvedeného v konfiguračním souboru IBM MQ MQI client

Eventuálně můžete přepsat atribut v konfiguračním souboru IBM MQ MQI client zadáním vlastnosti jako systémové vlastnosti v příkazu **java** . Chcete-li určit vlastnost jako systémovou vlastnost, použijte předchozí formát.

Pouze následující atributy v konfiguračním souboru IBM MQ MQI client jsou relevantní pro IBM MQ classes for JMS. Pokud uvedete nebo přepisují jiné atributy, nemá žádný efekt. Konkrétně si všimněte, že `ChannelDefinitionFile` a `ChannelDefinitionDirectory` v sekci stanza CHANNELS konfiguračního souboru klienta nejsou použity. Podrobné informace o použití tabulky CCDT s produktem IBM MQ classes for JMSnaleznete v příručce “Použití tabulky definic kanálů klienta s IBM MQ classes for JMS” na stránce 252 .

Tabulka 9. Který oddíl konfiguračního souboru klienta obsahuje který atribut

sekce	**Atribut**
stanza CHANNELS konfiguračního souboru klienta	Put1DefaultAlwaysSync
stanza CHANNELS konfiguračního souboru klienta	DefRecon
stanza CHANNELS konfiguračního souboru klienta	ReconDelay
stanza CHANNELS konfiguračního souboru klienta	PasswordProtection

Tabulka 9. Který oddíl konfiguračního souboru klienta obsahuje který atribut (pokračování)	
sekce	**Atribut**
ClientExitSekce Cesta ke konfiguračnímu souboru klienta	ExitsDefaultPath
ClientExitSekce Cesta ke konfiguračnímu souboru klienta	ExitsDefaultPath64
ClientExitSekce Cesta ke konfiguračnímu souboru klienta	JavaExitsClasspath
stanza JMQI konfiguračního souboru klienta	useMQCSPauthentication
stanzaMessageBuffer konfiguračního souboru klienta	MaximumSize
stanzaMessageBuffer konfiguračního souboru klienta	PurgeTime
stanzaMessageBuffer konfiguračního souboru klienta	UpdatePercentage
Sekce TCP konfiguračního souboru klienta	ClntRcvBufSize
Sekce TCP konfiguračního souboru klienta	ClntSndBufSize
Sekce TCP konfiguračního souboru klienta	Časový limit připojení
Sekce TCP konfiguračního souboru klienta	KeepAlive

Další podrobnosti o konfiguraci produktu IBM MQ MQI client naleznete v tématu Konfigurace klienta pomocí konfiguračního souboru .

stanza Java Standardní trasování prostředí
Pomocí oddílu Nastavení trasování standardního prostředí produktu Java lze konfigurovat prostředek trasování produktu IBM MQ classes for JMS .

com.ibm.msg.client.commonservices.trace.outputName = *traceOutputNázev*
traceOutputName je adresář a název souboru, do kterého je odeslán výstup trasování.

Při výchozím nastavení jsou trasovací informace zapsány do trasovacího souboru v aktuálním pracovním adresáři aplikace. Název trasovacího souboru závisí na prostředí, v němž je aplikace spuštěna:

- Pro IBM MQ classes for JMS for IBM MQ 9.0.0 Fix Pack 1 nebo starší, je trasování zapsáno do souboru s názvem `mqjms_%PID%.trc`.
- V 9.0.0.2 Pokud v produktu IBM MQ 9.0.0 Fix Pack 2byla aplikace načtena ze souboru JAR `com.ibm.mqjms.jar`do souboru IBM MQ classes for JMS , je trasování zapsáno do souboru s názvem `mqjava_%PID%.trc`.
- V 9.0.0.2 Pokud v produktu IBM MQ 9.0.0 Fix Pack 2byla aplikace načtena IBM MQ classes for JMS z přemístitelného souboru JAR `com.ibm.mq.allclient.jar`, je trasovací soubor zapsán do souboru s názvem `mqjavaclient_%PID%.trc`.
- V 9.0.0.10 Pokud v produktu IBM MQ 9.0.0 Fix Pack 10byla aplikace načtena ze souboru JAR `com.ibm.mqjms.jar`, je zapsána do souboru s názvem `mqjava_%PID%.cl%u.trc`, pokud aplikace má hodnotu IBM MQ classes for JMS .
- V 9.0.0.10 Pokud v produktu IBM MQ 9.0.0 Fix Pack 10byla aplikace načtena IBM MQ classes for JMS z přemístitelného souboru JAR `com.ibm.mq.allclient.jar`, je trasovací soubor zapsán do souboru s názvem `mqjavaclient_%PID%.cl%u.trc`.

kde *%PID%* je identifikátor procesu trasované aplikace, a *%u* je jedinečné číslo pro rozlišení souborů mezi podprocesy spuštěnými trasováním v různých zavaděčích tříd Java.

Uvedete-li alternativní adresář, musí existovat a musíte mít oprávnění k zápisu do tohoto adresáře. Nemáte-li oprávnění k zápisu, bude výstup trasování zapsán do `System.err`.

com.ibm.msg.client.commonservices.trace.include = *includeList*
includeList je seznam balíků a tříd, které jsou trasovány, nebo speciální hodnoty `ALL` nebo `NONE`.

Oddělte názvy balíků nebo tříd středníkem, `;`. *includeList* standardně zobrazuje `ALL`a trasuje všechny balíky a třídy v IBM MQ classes for JMS.

Poznámka: Můžete zahrnout balík, ale pak vyloučit podbalíky tohoto balíku. Pokud například zahrnete balík `a.b` a vyloučíte balík `a.b.x`, trasování zahrnuje vše v `a.b.y` a `a.b.z`, ale ne `a.b.x` nebo `a.b.x.1`.

com.ibm.msg.client.commonservices.trace.exclude = *excludeList*
excludeList je seznam balíků a tříd, které nejsou trasovány, nebo speciální hodnoty `ALL` nebo `NONE`.

Oddělte názvy balíků nebo tříd středníkem, `;`. *excludeList* standardně zobrazuje `NONE`, a proto vylučuje žádné balíky a třídy v IBM MQ classes for JMS , které jsou trasovány.

Poznámka: Balík můžete vyloučit, ale pak zahrnout podbalíky tohoto balíku. Například, pokud vyloučíte balík `a.b` a zahrnete balík `a.b.x`, trasování zahrnuje vše v `a.b.x` a `a.b.x.1`, ale ne `a.b.y` nebo `a.b.z`.

Zahrne se jakýkoli balík nebo třída, která je uvedená, na stejné úrovni, jak je zahrnuta i vyloučená.

com.ibm.msg.client.commonservices.trace.maxBytes = *maxArrayBajty*
maxArrayBytes je maximální počet bajtů, které jsou trasovány z libovolných bajtových polí.

Je-li hodnota *maxArrayBytes* nastavena na kladné celé číslo, omezuje počet bajtů v bajtovém poli, které jsou zapsány do trasovacího souboru. Ořízne bajtové pole po zapsání *maxArrayBytes* ven. Nastavení *maxArrayBytes* snižuje velikost výsledného trasovacího souboru a snižuje účinek trasování na výkon aplikace.

Hodnota `0` této vlastnosti znamená, že žádný obsah žádného bajtového pole není odeslán do trasovacího souboru.

Předvolená hodnota je `-1`, která odstraňuje jakékoli omezení počtu bajtů v bajtovém poli, které se posílají do trasovacího souboru.

com.ibm.msg.client.commonservices.trace.limit = *maxTraceBajty*
maxTraceBytes je maximální počet bajtů, které jsou zapsány do výstupního trasovacího souboru.

Produkt *maxTraceBytes* pracuje s produktem *traceCycles*. Pokud se počet bajtů trasování nachází v blízkosti limitu, soubor se zavře a spustí se nový výstupní soubor trasování.

Hodnota `0` znamená, že výstupní trasovací soubor má nulovou délku. Předvolená hodnota je `-1`, což znamená, že množství dat, která mají být zapsána do výstupního souboru trasování, je neomezeno.

com.ibm.msg.client.commonservices.trace.count = *traceCycles*
traceCycles je počet výstupních souborů trasování, které se mají procházet.

Pokud aktuální trasovací výstupní soubor dosáhne limitu zadaného pomocí *maxTraceBytes*, soubor se zavře. Další výstup trasování se zapisuje do dalšího výstupního souboru trasování v pořadí. Každý trasovací výstupní soubor se odlišuje od číselné přípony připojené k názvu souboru. Aktuální nebo poslední trasovací výstupní soubor je `mqjms.trc.0`, další nejnovější trasovací výstupní soubor je `mqjms.trc.1`. Starší trasovací soubory postupují podle stejného vzoru číslování až do limitu.

Standardní hodnota *traceCycles* je 1. Je-li hodnota *traceCycles* 1, dosáhne-li aktuální výstupní soubor trasování jeho maximální velikost, bude soubor uzavřen a odstraněn. Je spuštěn nový výstupní soubor trasování se stejným názvem. Proto v daném okamžiku existuje pouze jeden výstupní soubor trasování.

com.ibm.msg.client.commonservices.trace.parameter = *traceParameters*
Produkt *traceParameters* řídí, zda jsou parametry metod a návratové hodnoty zahrnuty do trasování.

Výchozí hodnota parametru *traceParameters* je TRUE. Je-li parametr *traceParameters* nastaven na hodnotu FALSE, trasují se pouze podpisy metody.

com.ibm.msg.client.commonservices.trace.startup = *spuštění*
Existuje inicializační fáze IBM MQ classes for JMS , během které jsou prostředky alokovány. Hlavní prostředek trasování je inicializován během fáze přidělování prostředků.

Je-li parametr *startup* nastaven na hodnotu TRUE, je použito trasování spouštění. Informace o trasování se vytvoří okamžitě a zahrnují nastavení všech komponent včetně samotného trasovacího prostředku. Informace o trasování spuštění lze použít k diagnostice problémů s konfigurací. Informace o trasování při spuštění se vždy zapisují do `System.err`.

Výchozí hodnota parametru *startup* je FALSE.

startup je zkontrolováno před dokončením inicializace. Z tohoto důvodu specifikujte pouze vlastnost na příkazovém řádku jako systémovou vlastnost Java . Neuvádějte jej do konfiguračního souboru IBM MQ classes for JMS .

com.ibm.msg.client.commonservices.trace.compress = *compressedTrace*
Chcete-li komprimovat trasovací výstup, nastavte *compressedTrace* na TRUE .

Standardní hodnota *compressedTrace* je FALSE.

Je-li parametr *compressedTrace* nastaven na hodnotu TRUE, je výstup trasování komprimován. Výchozí název výstupního souboru trasování má příponu `.trz`. Je-li komprese nastavena na FALSE, výchozí hodnota, má soubor příponu `.trc` , aby indikoval, že je nekomprimovaný. Pokud však název souboru pro výstup trasování byl zadán v souboru *traceOutputName* , bude místo něj použit název. Pro tento soubor se nepoužije žádná přípona.

Komprimovaný výstup trasování je menší než nekomprimovaný. Protože je zde méně vstupů/výstupů, lze jej zapsat rychleji než nekomprimované trasování. Komprimované trasování má menší vliv na výkon IBM MQ classes for JMS než nekomprimované trasování.

Je-li nastavena hodnota *maxTraceBytes* a *traceCycles* , vytvoří se více komprimovaných trasovacích souborů místo několika prostých textových souborů.

Pokud IBM MQ classes for JMS končí neřízeným způsobem, komprimovaný trasovací soubor nemusí být platný. Z tohoto důvodu musí být komprese trasování použita pouze v případě, že se IBM MQ classes for JMS vypíná řízeným způsobem. Kompresi trasování používejte pouze v případě, že problémy, které jsou předmětem šetření, nezpůsobují neočekávané zastavení prostředí JVM. Nepoužívejte kompresi trasování při diagnostice problémů, které mohou způsobit `System.Halt()` ukončení činnosti nebo nestandardní, nekontrolované ukončení prostředí JVM.

com.ibm.msg.client.commonservices.trace.level = *traceLevel*
traceLevel uvádí úroveň filtrování pro trasování. Definovaná úroveň trasování je následující:

- `TRACE_NONE`: `0`
- `TRACE_EXCEPTION`: `1`
- `TRACE_WARNING`: `3`
- `TRACE_INFO`: `6`
- `TRACE_ENTRYEXIT`: `8`
- `TRACE_DATA`: `9`
- `TRACE_ALL`: `Integer.MAX_VALUE`

Každá úroveň trasování zahrnuje všechny nižší úrovně. Je-li například úroveň trasování nastavena na hodnotu `TRACE_INFO`, je do trasování zapisován libovolný trasovací bod s definovanou úrovní `TRACE_EXCEPTION`, `TRACE_WARNING`nebo `TRACE_INFO` . Všechny ostatní stopové body jsou vyloučeny.

com.ibm.msg.client.commonservices.trace.standalone = *standaloneTrace*

Produkt *standaloneTrace* řídí, zda je služba trasování klienta produktu IBM MQ JMS použita v prostředí produktu WebSphere Application Server .

Je-li parametr *standaloneTrace* nastaven na hodnotu TRUE, jsou pro určení konfigurace trasování použity vlastnosti trasování klienta produktu IBM MQ JMS .

Je-li parametr *standaloneTrace* nastaven na hodnotu FALSEa klient produktu IBM MQ JMS je spuštěn v kontejneru WebSphere Application Server , použije se trasovací služba WebSphere Application Server . Trasovací informace, které se vygenerují, závisí na nastavení trasování aplikačního serveru.

Standardní hodnota *standaloneTrace* je FALSE.

Sekce protokolování
Použijte sekci Logging ke konfiguraci protokolovacího zařízení IBM MQ classes for JMS .

Do oddílu Protokolování lze zahrnout následující vlastnosti:

com.ibm.msg.client.commonservices.log.outputName = cesta
Název souboru protokolu používaného protokolovacím zařízením produktu IBM MQ classes for JMS . Výchozí hodnota je `mqjms.log`, která se zapisuje do aktuálního pracovního adresáře prostředí Java Runtime Environment, ve kterém je spuštěn produkt IBM MQ classes for JMS .

Vlastnost může mít jednu z následujících hodnot:

- Název jedné cesty
- Čárkami oddělený seznam úplných názvů (všechna data jsou protokolována do všech souborů)

Každý název cesty může být absolutní nebo relativní cesta nebo název:

"stderr" nebo "System.err"
Představuje standardní chybový proud.

"stdout" nebo "System.out"
Představuje standardní výstupní proud.

com.ibm.msg.client.commonservices.log.maxBytes
Maximální počet bajtů, které se zaprotokolují z jakéhokoli volání do dat zpráv protokolu.

Kladné celé číslo
Data jsou zapsána až do této hodnoty počtu bajtů na protokol volání.

0
Žádná data nejsou zapsána.

-1
Neomezená data jsou zapsána (výchozí).

com.ibm.msg.client.commonservices.log.limit
Maximální počet bajtů, které jsou zapsány do libovolného souboru protokolu 1 (výchozí je 262144).

Kladné celé číslo
Data jsou zapsána až do této hodnoty bajtů na soubor protokolu.

0
Žádná data nejsou zapsána.

-1
Neomezená data jsou zapsána.

com.ibm.msg.client.commonservices.log.count
Počet souborů protokolu, které se mají cyklině procházet. Protože každý soubor dosáhne trasování `com.ibm.msg.client.commonservices.trace.limit` , začne se v dalším souboru, výchozí hodnota je 3.

Kladné celé číslo
Počet souborů k cyklu.

0
Jediný soubor.

stanza Java SE Specifics
Použijte sekci Java SE Specifics ke konfiguraci vlastností, které se používají, když se IBM MQ classes for JMS používá v prostředí Java Standard Edition .

com.ibm.msg.client.commonservices.j2se.produceJavaCore = TRUE|FALSE
Určuje, zda je soubor jádra Javazapsán okamžitě poté, co IBM MQ classes for JMS vygeneroval soubor FDC. Je-li tato hodnota nastavena na hodnotu TRUE, je soubor core produktu Javavytvořen v pracovním adresáři prostředí Java Runtime Environment, ve kterém je spuštěn produkt IBM MQ classes for JMS .

TRUE
Generujte JavaCore, s výhradou schopnosti prostředí Java Runtime Environment to udělat.

FALSE
Negenerovat jádro Java; jedná se o výchozí hodnotu.

Sekce Vlastnosti IBM MQ
Použijte sekci Vlastnosti IBM MQ k nastavení vlastností, které ovlivňují způsob interakce produktu IBM MQ classes for JMS s produktem IBM MQ.

com.ibm.msg.client.wmq.compat.base.internal.MQQueue.smallMsgsBufferReductionThreshold
Pokud se aplikace používající produkt IBM MQ classes for JMS připojuje ke správci front produktu IBM MQ pomocí režimu migrace poskytovatele systému zpráv produktu IBM MQ , bude při příjmu zpráv IBM MQ classes for JMS používat výchozí velikost vyrovnávací paměti 4 kB. Pokud se zpráva, kterou se aplikace pokouší dostat, je větší než 4 kB, program IBM MQ classes for JMS změní velikost vyrovnávací paměti tak, aby byla dostatečně velká, aby pojmula zprávu. Větší velikost vyrovnávací paměti se pak použije při přijetí následných zpráv.

Tato vlastnost řídí, kdy je velikost vyrovnávací paměti redukována zpět na 4 kB. Při výchozím nastavení je velikost vyrovnávací paměti snížena o 4 kB, je-li přijato deset po sobě jdoucích zpráv, které jsou menší než větší velikost vyrovnávací paměti. Chcete-li obnovit velikost vyrovnávací paměti zpět na 4 kB při každém přijetí zprávy, nastavte vlastnost na hodnotu 0.

0

Vyrovnávací paměť se vždy resetuje na výchozí velikost.

10

Toto je výchozí hodnota. Velikost vyrovnávací paměti bude změněna po desáté zprávě.

com.ibm.msg.client.wmq.receiveConversionCCSID
Pokud se aplikace používající produkt IBM MQ classes for JMS připojuje ke správci front produktu IBM MQ s použitím běžného režimu poskytovatele systému zpráv produktu IBM MQ , lze vlastnost `receiveConversionCCSID` nastavit tak, aby přepsala výchozí hodnotu CCSID ve struktuře MQMD, která se používá pro příjem zpráv od správce front. MQMD standardně obsahuje pole CCSID nastavené na hodnotu 1208, lze jej však změnit například tehdy, pokud správce front nemůže převést zprávy na tuto kódovou stránku.

Platné hodnoty jsou libovolné platné číslo CCSID nebo jedna z následujících hodnot:

-1
Použijte výchozí nastavení platformy.

1208
Toto je výchozí hodnota.

stanza Client-mode specifics
Pomocí stanzy Client-mode specifikujte vlastnosti, které se použijí, když se produkt IBM MQ classes for JMS připojí ke správci front, který používá přenos CLIENT.

com.ibm.mq.polling.RemoteRequestEntry
Uvádí interval zjišťování, který IBM MQ classes for JMS používá ke kontrole přerušených připojení, když čeká na odpověď od správce front.

Kladné celé číslo
Počet milisekund čekání před kontrolou. Výchozí hodnota je 10000 nebo 10 sekund. Minimální hodnota je 3000 a nižší hodnoty jsou zpracovávány stejným způsobem jako minimální hodnota.

Vlastnosti použité ke konfiguraci chování klienta produktu JMS
Tyto vlastnosti slouží ke konfiguraci chování klienta produktu JMS .

com.ibm.mq.jms.SupportMQExtensions TRUE|FALSE
Specifikace JMS 2.0 zavádí změny v chování některých způsobů chování. Produkt IBM MQ 8.0 obsahuje vlastnost `com.ibm.mq.jms.SupportMQExtensions`, kterou lze nastavit na hodnotu *TRUE*, aby se tato změněná chování vrátila zpět k předchozím implementacím. Vrácení změněného chování může být nezbytné u aplikací produktu JMS 2.0 a také u některých aplikací, které používají rozhraní API produktu JMS 1.1 , ale spouštějí se na serveru IBM MQ 8.0 IBM MQ classes for JMS.

PRAVDA
Následující tři oblasti funkčnosti jsou převedeny zpět nastavením volby SupportMQExtensions na hodnotu TRUE:

Priorita zprávy
Pro zprávy může být přiřazena priorita, *0* - *9*. Před JMS 2.0mohou zprávy také použít hodnotu *-1*označující, že je použita výchozí priorita fronty. JMS 2.0 nepovoluje nastavení priority zprávy *-1* . Zapnutí `SupportMQExtensions` umožňuje použít hodnotu *-1* .

ID klienta
Specifikace JMS 2.0 vyžaduje, aby ID klientů bez hodnoty null byla při navázání spojení kontrolována na jedinečnost. Zapnutí `SupportMQExtensions`znamená, že tento požadavek není ignorován a že ID klienta lze použít znovu.

NoLocal
Specifikace JMS 2.0 vyžaduje, aby při zapnutí této konstanty nemohl spotřebitel přijímat zprávy, které jsou publikovány se stejným ID klienta. Před JMS 2.0byl tento atribut nastaven na odběrateli, aby zabránil příjmu zpráv, které jsou publikovány vlastním připojením. Zapnutí produktu `SupportMQExtensions` vrátí toto chování na její předchozí implementaci.

NEPRAVDA
Změny chování jsou zachovány.

com.ibm.msg.client.jms.ByteStreamReadOnlyAfterSend= TRUE|FALSE
V produktu IBM MQ 8.0.0 Fix Pack 2může produkt IBM MQ classes for JMS po odeslání zprávy o bajtech nebo proudu nastavit stav zprávy, která byla právě odeslána pouze ke čtení, nebo pouze pro zápis.

PRAVDA
Objekty jsou nastaveny na čtení pouze po odeslání. Nastavení této hodnoty zachovává kompatibilitu se specifikací JMS 2.0

NEPRAVDA
Objekty jsou nastaveny na zápis pouze poté, co byly odeslány. Toto je výchozí hodnota.

Související pojmy
“Vlastnost SupportMQExtensions” na stránce 294
Specifikace JMS 2.0 zavádí změny v chování některých způsobů chování. Produkt IBM MQ 8.0 a novější zahrnuje vlastnost `com.ibm.mq.jms.SupportMQExtensions`, kterou lze nastavit na hodnotu *TRUE*, aby se tato změněná chování vrátila zpět na předchozí implementace.

Konfigurace STEPLIB pro IBM MQ classes for JMS v systému z/OS
V systému z/OSmusí STEPLIB použitá za běhu obsahovat knihovny IBM MQ SCSQAUTH a SCSQANLE. Tyto knihovny zadejte ve spouštěcím souboru JCL nebo pomocí souboru `.profile` .

Ze systémových služeb produktu UNIX and Linux je možné přidat tyto řádky pomocí řádku ve vašem `.profile` , jak je znázorněno v následujícím úseku kódu, a nahradit `thlqual` kvalifikátorem datové sady vysoké úrovně, který jste zvolili při instalaci produktu IBM MQ:

```
export STEPLIB=thlqual.SCSQAUTH:thlqual.SCSQANLE:$STEPLIB
```

V jiných prostředích je zpravidla třeba upravit spouštěcí skript JCL, aby obsahoval SCSQAUTH a SCSQANLE na zřetězení STEPLIB:

```
STEPLIB DD DSN=thlqual.SCSQAUTH,DISP=SHR
        DD DSN=thlqual.SCSQANLE,DISP=SHR
```

IBM MQ classes for JMS a nástroje pro správu softwaru
Nástroje pro správu softwaru, jako např. Apache Maven, lze použít s produktem IBM MQ classes for JMS.

Řada velkých rozvojových organizací používá tyto nástroje k centrální správě úložišť knihoven jiných dodavatelů.

IBM MQ classes for JMS se skládá z určitého počtu souborů JAR. Když vyvíjíte jazykové aplikace produktu Java pomocí tohoto rozhraní API, je na počítači, na kterém je vyvíjována aplikace, vyžadována instalace buď produktu IBM MQ Server, klienta IBM MQ , nebo klienta IBM MQ SupportPac .

Chcete-li použít takový nástroj a přidat soubory JAR, které tvoří IBM MQ classes for JMS do centrálně spravovaného úložiště, musí být dodrženy následující body:

- Úložiště nebo kontejner musí být k dispozici pouze pro vývojáře v rámci vaší organizace. Jakékoli rozdělení mimo organizaci není povoleno.
- Úložiště musí obsahovat úplnou a konzistentní sadu souborů JAR z jedné verze produktu IBM MQ nebo z opravné sady Fix Pack.
- Jste zodpovědní za aktualizaci úložiště pomocí jakékoli údržby poskytované produktem IBM Support.

V produktu IBM MQ 8.0musí být do úložiště nainstalovány následující soubory JAR:

- `com.ibm.mq.allclient.jar`.
- Je-li použit produkt IBM MQ classes for JMS, je vyžadován produkt `jms.jar` .
- Produkt `fscontext.jar` je vyžadován, pokud používáte produkt IBM MQ classes for JMS a přistupujete ke spravovaným objektům produktu JMS , které jsou uloženy v kontextu rozhraní JNDI systému souborů.
- `providerutil.jar` , pokud používáte produkt IBM MQ classes for JMS a přistupujete ke spravovaným objektům produktu JMS , které jsou uloženy v kontextu JNDI systému souborů.

V produktu IBM MQ 9.0jsou vyžadovány soubory JAR Bouncy Castle Security a soubory JAR podpory CMS. Další informace viz “Co je nainstalováno pro IBM MQ classes for JMS” na stránce 74 a Podpora pro jiná prostředí nežIBM JRE.

Spuštění aplikací produktu IBM MQ classes for JMS v rámci struktury Java Security Manager

Produkt IBM MQ classes for JMS může být spuštěn se zapnutým správcem zabezpečení produktu Java . Chcete-li úspěšně spustit aplikace se zapnutým serverem Java Security Manager , musíte nakonfigurovat prostředí Java virtual machine (JVM) s vhodným konfiguračním souborem zásad.

Nejjednodušším způsobem, jak vytvořit vhodný definiční soubor zásad, je změnit konfigurační soubor zásad dodaný spolu s prostředím Java runtime environment (JRE). U většiny systémů se tento soubor nachází v adresáři `lib/security/java.policy` relativně vzhledem k adresáři JRE. Konfigurační soubor zásad můžete upravit buď pomocí svého preferovaného editoru, nebo pomocí programu nástroje zásad dodaného s vaším prostředím JRE.

Důležité:

Termín *allowlist* nahradil termín *whitelist*. V případě produktu IBM MQ 9.0 a novějších vydání toto zahrnuje názvy systémových vlastností produktu Java zmíněné v tomto tématu (**com.ibm.mq.jms.***). Nemusíte měnit žádnou existující konfiguraci. Předchozí názvy systémové vlastnosti budou také pracovat.

Použijete-li ke své aplikaci mechanizmus Java Security Manager , musíte udělit následující oprávnění:

- FilePermission na libovolném souboru allowlist, který používáte, s oprávněním ke čtení pro režim ENFORCEMENT, zapište oprávnění pro režim DISCOVER.
- PropertyPermission (čtení) na vlastnostech **com.ibm.mq.jms.allowlist**, **com.ibm.mq.jms.allowlist.discover**a **com.ibm.mq.jms.allowlist.mode** .

Pro produkt Continuous Deliveryje vlastnost allowlisting ClassName podporována z produktu IBM MQ 9.0.1. Další informace viz “Povolit koncepce” na stránce 110.

V 9.0.0.1 Ve verzi produktu Long Term Support je ClassName allowlisting podporován s APAR IT14385a z IBM MQ 9.0.0 Fix Pack 1.

Příklad konfiguračního souboru zásad

Zde je příklad konfiguračního souboru zásad, který umožňuje, aby produkt IBM MQ classes for JMS byl úspěšně spuštěn pod výchozím správcem zabezpečení. Tento soubor bude třeba upravit, chcete-li uvést umístění určitých souborů a adresářů: *MQ_INSTALLATION_PATH* představuje adresář vysoké úrovně, ve kterém je produkt IBM MQ nainstalován, *MQ_DATA_DIRECTORY* představuje umístění datového adresáře MQ a *QM_NAME* je název správce front, pro který je přístup konfigurován.

```
grant codeBase "file:MQ_INSTALLATION_PATH/java/lib/*" {
  //We need access to these properties, mainly for tracing
  permission java.util.PropertyPermission "user.name","read";
  permission java.util.PropertyPermission "os.name","read";
  permission java.util.PropertyPermission "user.dir","read";
  permission java.util.PropertyPermission "line.separator","read";
  permission java.util.PropertyPermission "path.separator","read";
  permission java.util.PropertyPermission "file.separator","read";
  permission java.util.PropertyPermission "com.ibm.msg.client.commonservices.log.*","read";
  permission java.util.PropertyPermission "com.ibm.msg.client.commonservices.trace.*","read";
  permission java.util.PropertyPermission "Diagnostics.Java.Errors.Destination.Filename","read";
  permission java.util.PropertyPermission "com.ibm.mq.commonservices","read";
  permission java.util.PropertyPermission "com.ibm.mq.cfg.*","read";

  //Tracing - we need the ability to control java.util.logging
  permission java.util.logging.LoggingPermission "control";
  // And access to create the trace file and read the log file - assumed to be in the current
directory
  permission java.io.FilePermission "*","read,write";

  // We'd like to set up an mBean to control trace
  permission javax.management.MBeanServerPermission "createMBeanServer";
  permission javax.management.MBeanPermission "*","*";

  // We need to be able to read manifests etc from the jar files in the installation directory
  permission java.io.FilePermission "MQ_INSTALLATION_PATH/java/lib/-","read";

  //Required if mqclient.ini/mqs.ini configuration files are used
  permission java.io.FilePermission "MQ_DATA_DIRECTORY/mqclient.ini","read";
  permission java.io.FilePermission "MQ_DATA_DIRECTORY/mqs.ini","read";

  //For the client transport type.
  permission java.net.SocketPermission "*","connect,resolve";

  //For the bindings transport type.
  permission java.lang.RuntimePermission "loadLibrary.*";

  //For applications that use CCDT tables (access to the CCDT AMQCLCHL.TAB)
  permission java.io.FilePermission "MQ_DATA_DIRECTORY/qmgrs/QM_NAME/@ipcc/AMQCLCHL.TAB","read";

  //For applications that use User Exits
  permission java.io.FilePermission "MQ_DATA_DIRECTORY/exits/*","read";
  permission java.io.FilePermission "MQ_DATA_DIRECTORY/exits64/*","read";
  permission java.lang.RuntimePermission "createClassLoader";

  //Required for the z/OS platform
  permission java.util.PropertyPermission "com.ibm.vm.bitmode","read";
```

```
   // Used by the internal ConnectionFactory implementation
   permission java.lang.reflect.ReflectPermission "suppressAccessChecks";

   // Used for controlled class loading
   permission java.lang.RuntimePermission "setContextClassLoader";

   // Used to default the Application name in Client mode connections
   permission java.util.PropertyPermission "sun.java.command","read";

   // Used by the IBM JSSE classes
   permission java.util.PropertyPermission "com.ibm.crypto.provider.AESNITrace","read";

   //Required to determine if an IBM Java Runtime is running in FIPS mode,
   //and to modify the property values status as required.
   permission java.util.PropertyPermission "com.ibm.jsse2.usefipsprovider","read,write";
   permission java.util.PropertyPermission "com.ibm.jsse2.JSSEFIPS","read,write";
   //Required if an IBM FIPS provider is to be used for SSL communication.
   permission java.security.SecurityPermission "insertProvider.IBMJCEFIPS";

   // Required for non-IBM Java Runtimes that establish secure client
   // transport mode connections using mutual TLS authentication
   permission java.util.PropertyPermission "javax.net.ssl.keyStore","read";
   permission java.util.PropertyPermission "javax.net.ssl.keyStorePassword","read";
};
```

V tomto příkladě obsahuje příkaz `grant` oprávnění požadovaná příkazem IBM MQ classes for JMS. Chcete-li tyto příkazy udělit ve svém konfiguračním souboru zásad, může být nutné upravit názvy cest v závislosti na tom, kam jste nainstalovali produkt IBM MQ classes for JMS a kam ukládáte vaše aplikace.

Ukázkové aplikace dodávané s produktem IBM MQ classes for JMSa skripty pro jejich spuštění nepovolují správce zabezpečení.

Postup po instalaci pro aplikace produktu IBM MQ classes for JMS

Toto téma informuje o tom, jaké oprávnění aplikace IBM MQ classes for JMS potřebují, aby mohli přistupovat k prostředkům správce front. Také uvádí režimy připojení a popisuje, jak nakonfigurovat správce front, aby se aplikace mohly připojit v režimu klienta.

Nezapomeňte zkontrolovat soubor Readme produktu IBM MQ . Může obsahovat informace, které nahrazují informace v tomto tématu.

Objekty použité produktem JMS , které vyžadují autorizaci pro neprivilegované uživatele

Neprivilegovaní uživatelé potřebují udělit autorizaci pro přístup k frontám používaným produktem JMS. Každá aplikace JMS potřebuje autorizaci ke správci front, se kterým pracuje.

Podrobnosti o řízení přístupu v produktu IBM MQnajdete v tématu Nastavení zabezpečení.

Aplikace produktu IBM MQ classes for JMS potřebují k tomuto správci front oprávnění `connect` a `inq` . Můžete nastavit příslušná oprávnění pomocí obslužného příkazu **`setmqaut`** , například:

```
setmqaut -m QM1 -t qmgr -g jmsappsgroup +connect +inq
```

Pro doménu typu point-to-point jsou vyžadovány následující oprávnění:

- Fronty, které používají objekty MessageProducer , potřebují oprávnění `put` .
- Fronty, které používají objekty MessageConsumer a QueueBrowser , potřebují oprávnění `get`, `inq`a `browse` .
- Metoda QueueSession.createTemporaryQueue () potřebuje přístup k modelové frontě určené vlastností TEMPMODEL objektu továrny QueueConnection. Ve výchozím nastavení je tato modelová fronta SYSTEM.TEMP.MODEL.QUEUE.

Jsou-li některé z těchto front alias fronty, jejich cílové fronty vyžadují dotazové oprávnění. Je-li cílová fronta frontou klastru, vyžaduje také oprávnění k procházení.

For the publish/subscribe domain, the following queues are used if the IBM MQ classes for JMS are connecting to an IBM MQ queue manager in IBM MQ messaging provider migration mode:

- SYSTEM.JMS.ADMIN.QUEUE

- SYSTEM.JMS.REPORT.QUEUE
- SYSTEM.JMS.MODEL.QUEUE
- SYSTEM.JMS.PS.STATUS.QUEUE
- SYSTEM.JMS.ND.SUBSCRIBER.QUEUE
- SYSTEM.JMS.D.SUBSCRIBER.QUEUE
- SYSTEM.JMS.ND.CC.SUBSCRIBER.QUEUE
- SYSTEM.JMS.D.CC.SUBSCRIBER.QUEUE
- SYSTEM.BROKER.CONTROL.QUEUE

Další informace o režimu migrace poskytovatele systému zpráv produktu IBM MQ naleznete v tématu Konfigurace vlastnosti produktu JMS **PROVIDERVERSION** .

Navíc, pokud se produkt IBM MQ classes for JMS připojuje ke správci front v tomto režimu, každá aplikace, která publikuje zprávy, potřebuje přístup do fronty proudu určené továrnou TopicConnectionnebo objektem tématu. Při výchozím nastavení je tato fronta SYSTEM.BROKER.DEFAULT.STREAM.

Pokud použijete volbu ConnectionConsumer, IBM MQ Resource Adapter nebo poskytovatele systému zpráv WebSphere Application Server IBM MQ , může být zapotřebí další autorizace.

Fronty, které mají být načteny ConnectionConsumer , musí mít oprávnění `get`, `inqa` `browse` . Fronta nedoručených zpráv systému a fronta zpráv typu backend nebo fronta zpráv používaná serverem ConnectionConsumer musí mít oprávnění `put` a `passall` .

Když aplikace používá normální režim poskytovatele systému zpráv produktu IBM MQ k provádění publikování/odběru zpráv, aplikace využívá integrované funkce publikování/odběru poskytované správcem front. Informace o zabezpečení témat a front, které se používají, najdete v tématu Zabezpečení publikování a odběru .

Režimy připojení pro produkt IBM MQ classes for JMS
Aplikace produktu IBM MQ classes for JMS se může připojit ke správci front v režimu klienta nebo vazby. V režimu klienta se produkt IBM MQ classes for JMS připojuje ke správci front prostřednictvím protokolu TCP/IP. V režimu vázání se produkt IBM MQ classes for JMS připojuje přímo ke správci front pomocí rozhraní JNI (Java Native Interface).

Aplikace běžící v produktu WebSphere Application Server v produktu z/OS se může připojit ke správci front buď v rámci vazeb, nebo v režimu klienta, ale aplikace běžící v libovolném jiném prostředí v produktu z/OS se může připojit ke správci front pouze v režimu vazeb. Aplikace spuštěná na libovolné jiné platformě se může připojit ke správci front buď v rámci vazeb, nebo v režimu klienta.

Můžete použít aktuální nebo jakoukoli dřívější podporovanou verzi produktu IBM MQ classes for JMS s aktuálním správcem front a můžete použít aktuální nebo dřívější podporovanou verzi správce front s aktuální verzí produktu IBM MQ classes for JMS. Pokud směšujete různé verze, funkce je omezena na úroveň předchozí verze.

Následující části popisují podrobněji každý režim připojení.

Režim klienta

Chcete-li se připojit ke správci front v režimu klienta, může být aplikace IBM MQ classes for JMS spuštěna na stejném systému, v němž je spuštěn správce front, nebo na jiném systému. V každém případě se produkt IBM MQ classes for JMS připojí ke správci front prostřednictvím protokolu TCP/IP.

Režim vazeb

Chcete-li se připojit ke správci front v režimu vazeb, musí být aplikace produktu IBM MQ classes for JMS spuštěna na stejném systému, v němž je spuštěn správce front.

Produkt IBM MQ classes for JMS se připojuje přímo ke správci front pomocí rozhraní JNI (Java Native Interface). Chcete-li použít přenos vazeb, produkt IBM MQ classes for JMS musí být spuštěn v prostředí,

které má přístup ke knihovnám nativního rozhraní produktu IBM MQ Java . Další informace naleznete v tématu “Konfigurace knihoven JNI (Java Native Interface)” na stránce 79 .

Produkt IBM MQ classes for JMS podporuje následující hodnoty pro *ConnectOption*:

- VAZBA MQCNO_FASTPATH_BINDING
- VAZBA MQCNO_STANDARD_BINDING
- CQCNO_SHARED_BINDING
- VAZBA MQCNO_ISOLATED_BINDING
- MQCNO_SERIALIZE_CONN_TAG_QSG
- MQCNO_RESTRICT_CONN_TAG_QSG
- MQCNO_SERIALIZE_CONN_TAG_Q_MGR
- MQCNO_RESTRICT_CONN_TAG_Q_MGR

Chcete-li změnit volby připojení používané produktem IBM MQ classes for JMS, upravte vlastnost továrny připojení CONNOPT.

Další informace o možnostech připojení naleznete v tématu “Připojení ke správci front pomocí volání MQCONNX” na stránce 707 .

Chcete-li použít přenos vazeb, musí být používané běhové prostředí produktu Java podporovat identifikátor CCSID (Coded Character Set Identifier) správce front, ke kterému se produkt IBM MQ classes for JMS připojuje.

Podrobnosti o tom, jak určit, které CCSID jsou podporovány běhovým prostředím Java , lze nalézt v IBM MQ FDC s ID sondy 21, které se vygenerovalo při použití tříd IBM MQ V7 pro Java nebo IBM MQ V7 pro JMS.

Konfigurace správce front tak, aby se aplikace produktu IBM MQ classes for JMS mohly připojit v režimu klienta
Chcete-li nakonfigurovat správce front tak, aby se aplikace produktu IBM MQ classes for JMS mohly připojit v režimu klienta, je třeba vytvořit definici kanálu připojení serveru a spustit modul listener.

Vytvoření definice kanálu připojení serveru

Na všech platformách můžete pomocí příkazu MQSC DEFINE CHANNEL vytvořit definici kanálu připojení serveru. Prohlédněte si následující příklad:

```
DEFINE CHANNEL(JAVA.CHANNEL) CHLTYPE(SVRCONN) TRPTYPE(TCP)
```

IBM i V systému IBM můžete místo toho použít příkaz CL CRTMQMCHL, jako v následujícím příkladu:

```
CRTMQMCHL CHLNAME(JAVA.CHANNEL) CHLTYPE(*SVRCN)
TRPTYPE(*TCP)
MQMNAME(QMGRNAME)
```

V tomto příkazu je *QMGRNAME* název vašeho správce front.

Windows Linux V systémech Linux a Windowsmůžete rovněž vytvořit definici kanálu připojení k serveru pomocí produktu IBM MQ Explorer.

z/OS V systému z/OS můžete použít operace a ovládací panely k vytvoření definice kanálu připojení k serveru.

Název kanálu (JAVA.CHANNEL v předchozích příkladech) musí být stejný jako název kanálu zadaný vlastností CHANNEL továrny připojení, kterou vaše aplikace používá pro připojení ke správci front. Výchozí hodnota vlastnosti CHANNEL je SYSTEM.DEF.SVRCONN.

Spuštění modulu listener

Je třeba spustit modul listener pro správce front, pokud ještě jeden není spuštěn.

Multi V systému Multiplatformymůžete pomocí příkazu MQSC START LISTENER spustit modul listener po prvním vytvoření objektu listeneru pomocí příkazu MQSC DEFINE LISTENER, jak je uvedeno v následujícím příkladu:

```
DEFINE LISTENER(LISTENER.TCP) TRPTYPE(TCP) PORT(1414)
START LISTENER(LISTENER.TCP)
```

z/OS V systému z/OSse používá pouze příkaz START LISTENER jako v následujícím příkladu, ale při spuštění modulu listener musí být před spuštěním modulu listener spuštěn adresní prostor inicializátoru kanálu:

```
START LISTENER TRPTYPE(TCP) PORT(1414)
```

IBM i V systému IBM imůžete také použít příkaz CL STRMQMLSR ke spuštění modulu listener, jak je uveden v následujícím příkladu:

```
STRMQMLSR PORT(1414) MQMNAME(QMGRNAME)
```

V tomto příkazu je *QMGRNAME* název vašeho správce front.

ULW V systému UNIX, Linux, and Windowsmůžete ke spuštění modulu listener použít také příkaz **runmqlsr** , jako v následujícím příkladu:

```
runmqlsr -t tcp -p 1414 -m QMgrName
```

V tomto příkazu je *QMgrName* název vašeho správce front.

Windows Linux V systémech Linux a Windowsmůžete také spustit modul listener pomocí produktu IBM MQ Explorer.

z/OS V systému z/OSmůžete také použít operace a ovládací panely ke spuštění modulu listener.

Číslo portu, na kterém listener naslouchá, musí být stejné jako číslo portu zadané vlastností PORT továrny připojení, kterou vaše aplikace používá pro připojení ke správci front. Výchozí hodnota vlastnosti PORT je 1414.

Dvoubodové IVT pro IBM MQ classes for JMS

Program testu IVT (point-to-point installation test) se dodává s IBM MQ classes for JMS. Program se připojuje ke správci front v obou vazbách nebo v režimu klienta, odesílá zprávu do fronty s názvem SYSTEM.DEFAULT.LOCAL.QUEUE, a poté přijme zprávu z fronty. Program může vytvořit a konfigurovat všechny objekty, které vyžaduje dynamicky za běhu, nebo může použít rozhraní JNDI k načtení spravovaných objektů z adresářové služby.

Spusťte test ověření instalace bez použití rozhraní JNDI jako první, protože test je samostatný a nevyžaduje použití adresářové služby. Popis administrovaných objektů najdete v tématu Konfigurace objektů produktu JMS pomocí nástroje pro administraci.

Test ověření po pevné lince bez použití rozhraní JNDI.

V tomto testu program IVT vytváří a konfiguruje všechny objekty, které vyžaduje za běhu dynamicky za běhu a nepoužívá JNDI.

K dispozici je skript pro spuštění programu IVT. Skript se nazývá IVTRun na systémech UNIX and Linux a IVTRun.bat na systému Windowsa nachází se v podadresáři `bin` instalačního adresáře produktu IBM MQ classes for JMS .

Chcete-li spustit test v režimu vazeb, zadejte následující příkaz:

```
IVTRun -nojndi [-m qmgr ] [-v providerVersion ] [-t]
```

Chcete-li spustit test v režimu klienta, nejprve nastavte správce front tak, jak je popsáno v tématu “Konfigurace správce front pro příjem klientských připojení na platformách Multiplatforms” na stránce 1045 a všimněte si, že kanál, který má být použit, je standardně nastaven na hodnotu SYSTEM.DEF.SVRCONN a fronta, která má být použita, je SYSTEM.DEFAULT.LOCAL.QUEUE, poté zadejte následující příkaz:

```
IVTRun -nojndi -client -m qmgr -host hostname [-port port ] [-channel channel ]
[-v providerVersion ] [-ccsid ccsid ] [-t]
```

Na systémech z/OS není poskytnut žádný ekvivalentní skript, ale IVT můžete spustit v režimu vázání vyvoláním třídy Java přímo, pomocí následujícího příkazu:

```
java com.ibm.mq.jms.MQJMSIVT -nojndi [-m qmgr ] [-v providerVersion ] [-t]
```

Cesta ke třídě musí obsahovat com.ibm.mqjms.jar.

Parametry v příkazech mají následující význam:

-m *správce_front*
Název správce front, ke kterému se program IVT připojuje. Pokud spustíte test v režimu vazeb a vynecháte tento parametr, program IVT se připojí k výchozímu správci front.

-host *hostname*
Název hostitele nebo adresa IP systému, v němž je spuštěn správce front.

-port *port*
Číslo portu, na kterém naslouchá modul listener správce front. Výchozí hodnota je 1414.

-kanál *kanál*
Název kanálu MQI, který program IVT používá pro připojení ke správci front. Výchozí hodnota je `SYSTEM.DEF.SVRCONN`.

-v *providerVersion*
Úroveň vydání správce front, ke kterému se program IVT očekává připojit.

Tento parametr se používá k nastavení vlastnosti PROVIDERVERSION objektu továrny MQQueueConnectiona má stejné platné hodnoty jako vlastnosti PROVIDERVERSION. Další informace o tomto parametru proto, včetně jeho platných hodnot, viz JMS: changes to PROVIDERVERSION property a popis vlastnosti PROVIDERVERSION v Vlastnostech objektů IBM MQ classes for JMS.

Výchozí hodnota je `unspecified`.

-ccsid *CCSID*
Identifikátor (CCSID) kódované znakové sady nebo kódové stránky, který má být použit pro připojení. Výchozí hodnota je 819.

-t
Trasování je povoleno. Ve výchozím nastavení je trasování zakázáno.

Úspěšný test produkuje výstup podobný tomuto ukázkovému výstupu:

```
5724-H72, 5655-R36, 5724-L26, 5655-L82 (c) Copyright IBM Corp. 2008, 2023. All
Rights Reserved.
WebSphere MQ classes for Java(tm) Message Service 7.0
Installation Verification Test

Creating a QueueConnectionFactory
```

```
Creating a Connection
Creating a Session
Creating a Queue
Creating a QueueSender
Creating a QueueReceiver
Creating a TextMessage
Sending the message to SYSTEM.DEFAULT.LOCAL.QUEUE
Reading the message back again

Got message
JMSMessage class: jms_text
JMSType:     null
JMSDeliveryMode: 2
JMSExpiration:  0
JMSPriority:   4
JMSMessageID:   ID:414d5120514d5f6d627720202020202001edb14620005e03
JMSTimestamp:   1187170264000
JMSCorrelationID: null
JMSDestination:  queue:///SYSTEM.DEFAULT.LOCAL.QUEUE
JMSReplyTo:    null
JMSRedelivered:  false
JMSXUserID: mwhite
JMS_IBM_Encoding: 273
JMS_IBM_PutApplType: 28
JMSXAppID: IBM MQ Client for Java
JMSXDeliveryCount: 1
JMS_IBM_PutDate: 20070815
JMS_IBM_PutTime: 09310400
JMS_IBM_Format: MQSTR
JMS_IBM_MsgType: 8
A simple text message from the MQJMSIVT
Reply string equals original string
Closing QueueReceiver
Closing QueueSender
Closing Session
Closing Connection
IVT completed OK
IVT finished
```

Test ověření po pevné lince používající rozhraní JNDI.

V tomto testu program IVT používá JNDI k načtení spravovaných objektů z adresářové služby.

Před spuštěním testu musíte nakonfigurovat adresářovou službu, která je založena na serveru LDAP (Lightweight Directory Access Protocol) nebo na lokálním systému souborů. Musíte také nakonfigurovat administrativní nástroj produktu IBM MQ JMS , aby mohl používat adresářovou službu k ukládání spravovaných objektů. Další informace o těchto předpokladech naleznete v tématu “Nezbytné předpoklady pro IBM MQ classes for JMS” na stránce 72. Chcete-li získat informace o tom, jak nakonfigurovat administrativní nástroj produktu IBM MQ JMS , prohlédněte si téma Konfigurace nástroje pro administraci produktu JMS.

Program IVT musí být schopen použít rozhraní JNDI k načtení objektu továrny MQQueueConnectiona objektu MQQueue z adresářové služby. K dispozici je skript pro vytvoření těchto spravovaných objektů pro vás. Skript se nazývá IVTSetup na systémech UNIX and Linux a IVTSetup.bat na Windowsa nachází se v podadresáři `bin` instalačního adresáře IBM MQ classes for JMS . Chcete-li spustit skript, zadejte tento příkaz:

```
IVTSetup
```

Skript vyvolá administrativní nástroj produktu IBM MQ JMS pro vytvoření spravovaných objektů.

Objekt továrny MQQueueConnectionje svázán s názvem ivtQCF a je vytvořen s výchozími hodnotami pro všechny její vlastnosti, což znamená, že program IVT se spouští v režimu vazeb a připojuje se k výchozímu správci front. Chcete-li, aby se program IVT spustil v režimu klienta nebo se připojil ke správci front jinému, než je výchozí správce front, musíte použít administrativní nástroj produktu IBM MQ JMS nebo produkt IBM MQ Explorer , chcete-li změnit příslušné vlastnosti objektu továrny

MQQueueConnection. Informace o použití nástroje pro administraci produktu IBM MQ Explorer JMS naleznete v tématu Konfigurace objektů produktu JMS pomocí nástroje pro administraci. Informace o tom, jak používat produkt IBM MQ Explorernajdete v části Úvod do produktu IBM MQ Explorer nebo v nápovědě poskytované s produktem IBM MQ Explorer.

Objekt MQQueue je svázán s názvem ivtQ a je vytvořen s výchozími hodnotami pro všechny jeho vlastnosti, s výjimkou vlastností QUEUE, která má hodnotu SYSTEM.DEFAULT.LOCAL.QUEUE.

Když jste vytvořili spravované objekty, můžete spustit program IVT. Chcete-li spustit test pomocí rozhraní JNDI, zadejte následující příkaz:

```
IVTRun -url "providerURL" [-icf initCtxFact ] [-t]
```

Parametry v příkazu mají následující význam:

-url "*providerURL*"
Uniform resource locator (URL) adresářové služby. Adresa URL může mít jeden z následujících formátů:

- `ldap://hostname/contextName` , pro adresářovou službu založenou na serveru LDAP
- `file:/directoryPath` , pro adresářovou službu založenou na lokálním systému souborů

Všimněte si, že adresu URL musíte uzavřít do uvozovek (").

-icf *initCtxFakt*
Název třídy počáteční továrny kontextu, která musí mít jednu z následujících hodnot:

- `com.sun.jndi.ldap.LdapCtxFactory`, pro adresářovou službu založenou na serveru LDAP. Toto je výchozí hodnota.
- `com.sun.jndi.fscontext.RefFSContextFactory`, pro adresářovou službu založenou na lokálním systému souborů.

-t
Trasování je povoleno. Ve výchozím nastavení je trasování zakázáno.

Úspěšný test produkuje výstup podobný testu pro úspěšný test bez použití rozhraní JNDI. Hlavní rozdíl spočívá v tom, že výstup označuje, že test používá rozhraní JNDI k načtení objektu továrny MQQueueConnectiona objektu MQQueue.

I když to není nezbytně nutné, je dobrým zvykem po testu uklidit tím, že odstraníte spravované objekty vytvořené skriptem IVTSetup. Pro tento účel je k dispozici skript. Skript se nazývá IVTTidy v systémech UNIX and Linux a IVTTidy.bat na systému Windowsa nachází se v podadresáři `bin` instalačního adresáře produktu IBM MQ classes for JMS .

Určování problémů pro ověřovací test instalace typu point-to-point

Ověřovací test instalace může selhat z následujících důvodů:

- Pokud program IVT zapíše zprávu o tom, že nemůže najít třídu, zkontrolujte, zda je cesta ke třídě nastavena správně, jak je popsáno v tématu “Nastavení proměnných prostředí” na stránce 77.
- Test může selhat s následující zprávou:

```
Failed to connect to queue manager ' qmgr ' with connection mode ' connMode '
and host name ' hostname '
```

a s přidruženým kódem příčiny 2059. Proměnné ve zprávě mají následující význam:

QMGR
Název správce front, ke kterému se program IVT pokouší připojit. Tato vložená zpráva je prázdná, pokud se program IVT pokouší připojit k výchozímu správci front v režimu vazeb.

connMode
Režim připojení, který je buď `Bindings` , nebo `Client`.

název_hostitele
Název hostitele nebo adresa IP systému, v němž je spuštěn správce front.

Tato zpráva znamená, že správce front, ke kterému se program IVT pokouší připojit, není k dispozici. Zkontrolujte, zda je správce front spuštěn, a pokud se program IVT pokouší připojit k výchozímu správci front, ujistěte se, že správce front je definován jako výchozí správce front pro váš systém.

- Test může selhat s následující zprávou:

```
Failed to open MQ queue 'SYSTEM.DEFAULT.LOCAL.QUEUE'
```

Tato zpráva znamená, že fronta SYSTEM.DEFAULT.LOCAL.QUEUE neexistuje ve správci front, ke kterému je program IVT připojen. Jinak, pokud fronta existuje, program IVT nemůže otevřít frontu, protože není povolena pro vkládání a získávání zpráv. Zkontrolujte, zda fronta existuje a zda je povolena pro vkládání a získávání zpráv.

- Test může selhat s následující zprávou:

```
Unable to bind to object
```

Tato zpráva znamená, že existuje připojení k serveru LDAP, ale že server LDAP není správně nakonfigurován. Server LDAP buď není konfigurován pro ukládání objektů Java , nebo nejsou oprávnění na objektech nebo v příponě správná. Další nápověda v této situaci naleznete v dokumentaci k vašemu serveru LDAP.

- Test může selhat s následující zprávou:

```
The security authentication was not valid that was supplied for
QueueManager ' qmgr ' with connection mode 'Client' and host name ' hostname '
```

Tato zpráva znamená, že správce front není správně nastaven tak, aby přijímal připojení klienta ze systému. Podrobnosti viz “Konfigurace správce front pro příjem klientských připojení na platformách Multiplatforms” na stránce 1045.

Publikování/odběr IVT pro IBM MQ classes for JMS

Program IVT (publish/subscribe installation verification test) je dodáván s produktem IBM MQ classes for JMS. Program se připojí ke správci front buď v rámci vazeb, nebo v režimu klienta, přihlásí se k odběru tématu, publikuje zprávu na daném tématu a poté přijme zprávu, kterou právě publikoval. Program může vytvořit a konfigurovat všechny objekty, které vyžaduje dynamicky za běhu, nebo může použít rozhraní JNDI k načtení spravovaných objektů z adresářové služby.

Spusťte test ověření instalace bez použití rozhraní JNDI jako první, protože test je samostatný a nevyžaduje použití adresářové služby. Popis administrovaných objektů najdete v tématu Konfigurace objektů produktu JMS pomocí nástroje pro administraci.

Ověřovací test instalace typu publikování/odběr bez použití rozhraní JNDI

V tomto testu program IVT vytváří a konfiguruje všechny objekty, které vyžaduje za běhu dynamicky za běhu a nepoužívá JNDI.

K dispozici je skript pro spuštění programu IVT. Skript se nazývá PSIVTRun na systémech UNIX and Linux a PSIVTRun.bat na systému Windowsa nachází se v podadresáři `bin` instalačního adresáře produktu IBM MQ classes for JMS .

Chcete-li spustit test v režimu vazeb, zadejte následující příkaz:

```
PSIVTRun -nojndi [-m qmgr ] [-bqm brokerQmgr ] [-v providerVersion ] [-t]
```

Chcete-li spustit test v režimu klienta, nejprve nastavte správce front tak, jak je popsáno v tématu “Konfigurace správce front pro příjem klientských připojení na platformách Multiplatforms”

na stránce 1045 a všimněte si, že kanál, který má být použit, je standardně nastaven na hodnotu SYSTEM.DEF.SVRCONN, pak zadejte tento příkaz:

```
PSIVTRun -nojndi -client -m qmgr -host hostname [-port port ] [-channel channel ]
[-bqm brokerQmgr ] [-v providerVersion ] [-ccsid ccsid ] [-t]
```

Parametry v příkazech mají následující význam:

-m *správce_front*
Název správce front, ke kterému se program IVT připojuje. Pokud spustíte test v režimu vazeb a vynecháte tento parametr, program IVT se připojí k výchozímu správci front.

-host *hostname*
Název hostitele nebo adresa IP systému, v němž je spuštěn správce front.

-port *port*
Číslo portu, na kterém naslouchá modul listener správce front. Výchozí hodnota je 1414.

-kanál *kanál*
Název kanálu MQI, který program IVT používá pro připojení ke správci front. Výchozí hodnota je SYSTEM.DEF.SVRCONN.

-bqm *brokerQmgr*
Název správce front, v němž je zprostředkovatel spuštěn. Výchozí hodnota je název správce front, ke kterému se program IVT připojuje.

Tento parametr není relevantní pro číslo verze správce front v z 7 nebo vyšší.

-v *providerVersion*
Úroveň vydání správce front, ke kterému se program IVT očekává připojit.

Tento parametr se používá k nastavení vlastnosti PROVIDERVERSION objektu továrny MQTopicConnectiona má stejné platné hodnoty jako vlastnosti PROVIDERVERSION. Další informace o tomto parametru, včetně jeho platných hodnot, najdete v popisu vlastnosti PROVIDERVERSION v části Vlastnosti objektů IBM MQ classes for JMS.

Výchozí hodnota je unspecified.

-ccsid *CCSID*
Identifikátor (CCSID) kódované znakové sady nebo kódové stránky, který má být použit pro připojení. Výchozí hodnota je 819.

-t
Trasování je povoleno. Ve výchozím nastavení je trasování zakázáno.

Úspěšný test produkuje výstup podobný tomuto ukázkovému výstupu:

```
5724-H72, 5655-R36, 5724-L26, 5655-L82 (c) Copyright IBM Corp. 2008, 2023. All
Rights Reserved.
IBM MQ classes for Java(tm) Message Service 7.0
Publish/Subscribe Installation Verification Test

Creating a TopicConnectionFactory
Creating a Connection
Creating a Session
Creating a Topic
Creating a TopicPublisher
Creating a TopicSubscriber
Creating a TextMessage
Adding text
Publishing the message to topic://MQJMS/PSIVT/Information
Waiting for a message to arrive [5 secs max]...

Got message:
JMSMessage class: jms_text
JMSType:     null
JMSDeliveryMode: 2
JMSExpiration:  0
JMSPriority:   4
```

```
JMSMessageID:   ID:414d5120514d5f6d627720202020202001edb14620006706
JMSTimestamp:   1187182520203
JMSCorrelationID: ID:414d5120514d5f6d627720202020202001edb14620006704
JMSDestination:  topic://MQJMS/PSIVT/Information
JMSReplyTo:    null
JMSRedelivered:  false
JMSXUserID: mwhite
JMS_IBM_Encoding: 273
JMS_IBM_PutApplType: 26
JMSXAppID: QM_mbw
JMSXDeliveryCount: 1
JMS_IBM_PutDate: 20070815
JMS_IBM_ConnectionID: 414D5143514D5F6D627720202020202001EDB14620006601
JMS_IBM_PutTime: 12552020
JMS_IBM_Format: MQSTR
JMS_IBM_MsgType: 8
A simple text message from the MQJMSPSIVT program
Reply string equals original string
Closing TopicSubscriber
Closing TopicPublisher
Closing Session
Closing Connection
PSIVT finished
```

Ověřovací test instalace publikování/odběru s použitím rozhraní JNDI

V tomto testu program IVT používá JNDI k načtení spravovaných objektů z adresářové služby.

Před spuštěním testu musíte nakonfigurovat adresářovou službu, která je založena na serveru LDAP (Lightweight Directory Access Protocol) nebo na lokálním systému souborů. Musíte také nakonfigurovat administrativní nástroj produktu IBM MQ JMS , aby mohl používat adresářovou službu k ukládání spravovaných objektů. Další informace o těchto předpokladech naleznete v tématu “Nezbytné předpoklady pro IBM MQ classes for JMS” na stránce 72. Chcete-li získat informace o tom, jak nakonfigurovat administrativní nástroj produktu IBM MQ JMS , prohlédněte si téma Konfigurace nástroje pro administraci produktu JMS.

Program IVT musí být schopen použít rozhraní JNDI pro načtení objektu továrny MQTopicConnectiona objektu MQTopic z adresářové služby. K dispozici je skript pro vytvoření těchto spravovaných objektů pro vás. Skript se nazývá IVTSetup na systémech UNIX and Linux a IVTSetup.bat na Windowsa nachází se v podadresáři `bin` instalačního adresáře IBM MQ classes for JMS . Chcete-li spustit skript, zadejte tento příkaz:

```
IVTSetup
```

Skript vyvolá administrativní nástroj produktu IBM MQ JMS pro vytvoření spravovaných objektů.

Objekt továrny MQTopicConnectionje svázán s názvem ivtTCF a je vytvořen s výchozími hodnotami pro všechny její vlastnosti, což znamená, že se program IVT spouští v režimu vazeb, připojí se k výchozímu správci front a používá funkci vloženého publikování/odběru. Chcete-li, aby program IVT byl spuštěn v režimu klienta, připojte se ke správci front jinému než výchozímu správci front, nebo místo vestavěné funkce publikování/odběru použijte produkt IBM Integration Bus , je třeba pomocí nástroje pro administraci produktu IBM MQ JMS nebo Průzkumníka IBM MQ změnit příslušné vlastnosti objektu továrny MQTopicConnection. Informace o použití nástroje pro administraci produktu IBM MQ JMS naleznete v tématu Konfigurace objektů produktu JMS pomocí nástroje pro administraci. Informace o tom, jak používat program Průzkumník IBM MQ , naleznete v nápovědě dodávané s produktem IBM MQ Explorer.

Objekt MQTopic je svázán s názvem ivtT a je vytvořen s výchozími hodnotami pro všechny jeho vlastnosti, s výjimkou vlastnosti TOPIC, která má hodnotu MQJMS/PSIVT/Information.

Když jste vytvořili spravované objekty, můžete spustit program IVT. Chcete-li spustit test pomocí rozhraní JNDI, zadejte následující příkaz:

```
PSIVTRun -url "providerURL" [-icf initCtxFact ] [-t]
```

Parametry v příkazu mají následující význam:

-url "*providerURL*"
Uniform resource locator (URL) adresářové služby. Adresa URL může mít jeden z následujících formátů:

- `ldap://hostname/contextName` , pro adresářovou službu založenou na serveru LDAP
- `file:/directoryPath` , pro adresářovou službu založenou na lokálním systému souborů

Všimněte si, že adresu URL musíte uzavřít do uvozovek (").

-icf *initCtxFakt*
Název třídy počáteční továrny kontextu, která musí mít jednu z následujících hodnot:

- `com.sun.jndi.ldap.LdapCtxFactory`, pro adresářovou službu založenou na serveru LDAP. Toto je výchozí hodnota.
- `com.sun.jndi.fscontext.RefFSContextFactory`, pro adresářovou službu založenou na lokálním systému souborů.

-t
Trasování je povoleno. Ve výchozím nastavení je trasování zakázáno.

Úspěšný test produkuje výstup podobný testu pro úspěšný test bez použití rozhraní JNDI. Hlavní rozdíl spočívá v tom, že výstup označuje, že test používá rozhraní JNDI k načtení objektu továrny MQTopicConnectiona objektu MQTopic.

I když to není nezbytně nutné, je dobrým zvykem po testu uklidit tím, že odstraníte spravované objekty vytvořené skriptem IVTSetup. Pro tento účel je k dispozici skript. Skript se nazývá IVTTidy v systémech UNIX and Linux a IVTTidy.bat na systému Windowsa nachází se v podadresáři `bin` instalačního adresáře produktu IBM MQ classes for JMS .

Určování problémů pro ověřovací test instalace publikování/odběru

Ověřovací test instalace může selhat z následujících důvodů:

- Pokud program IVT zapíše zprávu o tom, že nemůže najít třídu, zkontrolujte, zda je cesta ke třídě nastavena správně, jak je popsáno v tématu “Nastavení proměnných prostředí” na stránce 77.
- Test může selhat s následující zprávou:

```
Failed to connect to queue manager ' qmgr ' with
connection mode ' connMode ' and host name ' hostname '
```

a s přidruženým kódem příčiny 2059. Proměnné ve zprávě mají následující význam:

QMGR
Název správce front, ke kterému se program IVT pokouší připojit. Tato vložená zpráva je prázdná, pokud se program IVT pokouší připojit k výchozímu správci front v režimu vazeb.

connMode
Režim připojení, který je buď `Bindings` , nebo `Client`.

název_hostitele
Název hostitele nebo adresa IP systému, v němž je spuštěn správce front.

Tato zpráva znamená, že správce front, ke kterému se program IVT pokouší připojit, není k dispozici. Zkontrolujte, zda je správce front spuštěn, a pokud se program IVT pokouší připojit k výchozímu správci front, ujistěte se, že správce front je definován jako výchozí správce front pro váš systém.

- Test může selhat s následující zprávou:

```
Unable to bind to object
```

Tato zpráva znamená, že existuje připojení k serveru LDAP, ale že server LDAP není správně nakonfigurován. Server LDAP buď není konfigurován pro ukládání objektů Java , nebo nejsou oprávnění na objektech nebo v příponě správná. Další nápověda v této situaci naleznete v dokumentaci k vašemu serveru LDAP.

- Test může selhat s následující zprávou:

```
The security authentication was not valid that was supplied for
QueueManager ' qmgr ' with connection mode 'Client' and host name ' hostname '
```

Tato zpráva znamená, že správce front není správně nastaven pro přijetí připojení klienta ze systému. Další informace viz “Konfigurace správce front pro příjem klientských připojení na platformách Multiplatforms” na stránce 1045.

Použití ukázkových aplikací produktu IBM MQ classes for JMS

Ukázkové aplikace produktu IBM MQ classes for JMS poskytují přehled běžných funkcí rozhraní API produktu JMS . Můžete je použít k ověření instalace a serveru systému zpráv, které vám pomohou při sestavování svých vlastních aplikací.

Informace o této úloze

Potřebujete-li pomoci při vytváření vlastních aplikací, můžete ukázkové aplikace použít jako výchozí bod. Pro každou aplikaci je k dispozici jak zdrojový, tak i kompilovaná verze. Přezkoumejte vzorový zdrojový kód a identifikujte klíčové kroky pro vytvoření každého požadovaného objektu pro vaši aplikaci (ConnectionFactory, Connection, Session, Destination, and a Producer, or a Consumer, nebo obojí) a pro nastavení libovolných specifických vlastností, které jsou potřeba k uvedení, jak chcete, aby vaše aplikace fungovala. Další informace viz “Zápis aplikací IBM MQ classes for JMS” na stránce 118. Ukázky mohou být předmětem změn v budoucích vydáních produktu IBM MQ.

Produkt Tabulka 10 na stránce 102 zobrazuje, kde jsou ukázkové aplikace produktu IBM MQ classes for JMS instalovány na jednotlivých platformách:

Tabulka 10. Instalační adresáře pro ukázkové aplikace produktu IBM MQ classes for JMS

Platforma	Adresář
UNIX UNIX Linux Linux	*MQ_INSTALLATION_PATH*/samp/jms/samples
Windows Windows	*MQ_INSTALLATION_PATH*\tools\jms\samples
IBM i IBM i	/qibm/proddata/mqm/java/samples/jms/samples
z/OS z/OS	*MQ_INSTALLATION_PATH*/java/samples/jms

V tomto adresáři jsou podadresáře, které obsahují jednu nebo více ukázkových aplikací, jak je uvedeno v tématu Tabulka 11 na stránce 102.

Tabulka 11. IBM MQ classes for JMS ukázkové aplikace

Název vzorku	Popis
JmsBrowser .java	Aplikace prohlížeče fronty JMS , která prohledává všechny dostupné zprávy ve jmenované frontě, aniž by je odstranila, v pořadí, v jakém by byla přijata odběratelskou aplikací.

Tabulka 11. IBM MQ classes for JMS ukázkové aplikace (pokračování)	
Název vzorku	**Popis**
JmsConsumer.java	Aplikace prohlížeče fronty produktu JMS , která hledá všechny dostupné zprávy v uvedené frontě, aniž by je odebrala, v pořadí, v jakém by byla přijata aplikací spotřebitele, vyhledáním instance továrny připojení a cílové instance v počátečním kontextu (Tato ukázka podporuje pouze kontext systému souborů).
JmsJndiConsumer.java	Aplikace odběratele produktu JMS (příjemce nebo odběratel), která přijímá zprávu z uvedeného místa určení (fronta nebo téma) vyhledáním instance továrny připojení a cílové instance v počátečním kontextu (Tato ukázka podporuje pouze kontext systému souborů).
JmsJndiProducer.java	Aplikace producenta produktu JMS (odesílatel nebo vydavatel), která odesílá jednoduchou zprávu do pojmenovaného místa určení (fronta nebo téma) vyhledáním instance továrny připojení a cílové instance v počátečním kontextu (Tato ukázka podporuje pouze kontext systému souborů).
JmsProducer.java	Producent produktu JMS (odesílatel nebo vydavatel), který odesílá jednoduchou zprávu do uvedeného místa určení (fronta nebo téma).
/interaktivní/	
SampleConsumerJava.java	Přijímat zprávy z tématu/fronty.
SampleProducerJava.java	Odeslat zprávu (zprávy) do tématu/fronty.
/interactive/helper/	
BaseOptions.java	Abstraktní třída, která může být rozšířena tak, aby poskytovala funkčnost (i) uživatele.
IsValidType.java	Abstraktní třída pro třídy kontroly platnosti.
JmsApp.java	Abstraktní třída, kterou lze rozšířit tak, aby poskytovala funkce spotřebitele/producenta.
Keys.java	Sada klíčů, které definují volby pro ukázkové aplikace.
Literals.java	Sada konstantních literálů.
MyContext.java.	Kontext, ve kterém jsou volby prezentovány.
Options.java	Poskytuje funkce pro volbu (y) uživatele.
OptionsPresenter.java	Kontext, ve kterém jsou prezentovány aktuální volby.
/simple/	
SimpleAsyncPutPTP.java.	Jednoduchá aplikace pro systém zpráv typu point-to-point, zpráva se odesílá asynchronně (také známá jako systém zpráv *fire-and-forget*). Nepřijímají se žádné zprávy.
SimpleDurableSub.java	Jednoduchá aplikace, která demonstruje službu trvalého odběru.

Tabulka 11. IBM MQ classes for JMS ukázkové aplikace (pokračování)	
Název vzorku	**Popis**
SimpleJNDILookup.java	Minimální a jednoduchá aplikace, která demonstruje vyhledávání objektů produktu JMS pomocí počátečního kontextu. Neprovede se žádné připojení ke správci front a žádné zprávy se neodesílají ani nepřijímají.
SimpleMQMDRead.java	Jednoduchá aplikace, která demonstruje, jak aplikace JMS může využít pole MQ Message Descriptor (MQMD) jako vlastnosti zprávy produktu JMS . Nejsou odeslány žádné zprávy; předpokládá se, že fronta v použití je naplněna některými zprávami.
SimpleMQMDWrite.java	Jednoduchá aplikace, která demonstruje, jak aplikace JMS může zapsat pole MQ Message Descriptor (MQMD). Nepřijímají se žádné zprávy.
SimplePTP.java.	Minimální a jednoduchá aplikace pro systém zpráv typu point-to-point.
SimplePubSub.java	Minimální a jednoduchá aplikace pro systém zpráv typu publikování-odběr.
SimpleReadAheadPTP.java	Jednoduchá aplikace pro systém zpráv typu point-to-point. Zprávy jsou odesílány ze správce front (také známý jako prostředek dopředného čtení). Nejsou odeslány žádné zprávy; předpokládá se, že fronta v použití je naplněna některými zprávami.
SimpleRequestor.java	Jednoduchá aplikace, která používá žadatele k odeslání zprávy s požadavkem a poté čeká na odpověď a přijímá odpověď. Pozn.: Předpokládá se, že některá jiná aplikace zpracuje zprávu požadavku a odešle zprávu odpovědi.
SimpleResponder.java.	Jednoduchá aplikace, která naslouchá cíli pro zprávu a odešle odpověď na místo určení zprávy replyTo . Aplikace je napsána pro práci ve spojení s ukázkou SimpleRequestor .
SimpleRetainedPub.java	Jednoduchá aplikace, která demonstruje zachované publikování. Nepřijímají se žádné zprávy.
SimpleWMQJMSPTP.java	Minimální a jednoduchá aplikace pro systém zpráv typu point-to-point.
SimpleWMQJMSPubSub.java	Minimální a jednoduchá aplikace pro publikování/odběr systému zpráv.

Produkt IBM MQ classes for JMS nabízí skript s názvem `runjms` , který lze použít ke spouštění ukázkových aplikací. Tento skript nastaví prostředí produktu IBM MQ , aby bylo možné spustit ukázkové aplikace produktu IBM MQ classes for JMS .

Tabulka 12 na stránce 104 zobrazuje umístění skriptu na každé platformě:

Tabulka 12. Umístění skriptu `runjms`	
Platforma	**Adresář**
UNIX UNIX Linux Linux	*MQ_INSTALLATION_PATH*`/java/bin/runjms`
Windows Windows	*MQ_INSTALLATION_PATH*`\java\bin\runjms.bat`

Tabulka 12. Umístění skriptu `runjms` (pokračování)	
Platforma	**Adresář**

IBM i	`/qibm/proddata/mqm/java/bin/runjms`
	, nebo
	`/qibm/proddata/mqm/java/bin/runjms64`

z/OS	*`MQ_INSTALLATION_PATH`*`java/bin/runjms`

Chcete-li použít skript `runjms` k vyvolání ukázkové aplikace, proveďte následující kroky:

Postup

1. Vyveďte příkazový řádek a přejděte do adresáře obsahujícího ukázkovou aplikaci, kterou chcete spustit.
2. Zadejte následující příkaz:

   ```
   Path to the runjms script/runjms sample_application_name
   ```

   Ukázková aplikace zobrazí seznam parametrů, které potřebuje.
3. Chcete-li spustit ukázku s těmito parametry, zadejte následující příkaz:

   ```
   Path to the runjms script/runjms sample_application_name parameters
   ```

Příklad

Linux Chcete-li například spustit ukázku JmsBrowser v systému Linux, zadejte následující příkazy:

```
cd /opt/mqm/samp/jms/samples
/opt/mqm/java/bin/runjms JmsBrowser -m QM1 -d LQ1
```

Související pojmy

“Co je nainstalováno pro IBM MQ classes for JMS” na stránce 74
Při instalaci produktu IBM MQ classes for JMSse vytvoří mnoho souborů a adresářů. V systému Windowsse při instalaci provádí některá konfigurace automaticky nastavováním proměnných prostředí. Na jiných platformách a v některých prostředích produktu Windows musíte nastavit proměnné prostředí před spuštěním aplikací produktu IBM MQ classes for JMS .

Skripty poskytnuté s IBM MQ classes for JMS

Je k dispozici řada skriptů, které pomáhají s běžnými úlohami, které je třeba provést při používání produktu IBM MQ classes for JMS.

Tabulka 13 na stránce 105 vypíše všechny skripty a jejich použití. Tyto skripty jsou umístěny v podadresáři `bin` instalačního adresáře produktu IBM MQ classes for JMS .

Tabulka 13. Skripty poskytnuté s IBM MQ classes for JMS	
Utility	**Použití**
Vyčištění [1]	Tento skript je udržován kvůli kompatibilitě s předchozími vydáními, ale neprovede žádnou funkci. Ruční vyčištění informací o odběru již není nezbytné
DefaultConfiguration	Spustí výchozí konfigurační aplikaci na platformách jiných než Windows.

Tabulka 13. Skripty poskytnuté s IBM MQ classes for JMS (pokračování)	
Utility	**Použití**
formatLog [1]	Tento skript je udržován kvůli kompatibilitě s předchozími vydáními, ale neprovede žádnou funkci. Výstup protokolu se nyní vytváří v čitelném textu.
IVTRun [1] IVTSetup [1] IVTTidy [1]	Používá se v testu ověření po pevné lince, jak je popsáno v tématu “Dvoubodové IVT pro IBM MQ classes for JMS” na stránce 94.
JMSAdmin [1]	Spustí administrativní nástroj produktu IBM MQ JMS , jak je popsáno v tématu Spuštění nástroje pro administraci.
JMSAdmin.config	Konfigurační soubor pro administrativní nástroj produktu IBM MQ JMS , jak je popsáno v tématu Konfigurace nástroje pro administraci produktu JMS.
PIVTRun [1]	Spustí testovací program pro ověření instalace publikování/odběru, jak je popsáno v tématu “Publikování/odběr IVT pro IBM MQ classes for JMS” na stránce 98.
PSReportDump.class	Tato třída je udržována kvůli kompatibilitě s předchozími vydáními, ale neprovádí žádnou funkci.
setjmsenv	Nastaví proměnné prostředí pro spuštění aplikace IBM MQ classes for JMS v 32bitovém Java virtuálním počítači (JVM) na systémech UNIX and Linux , jak je popsáno v “Nastavení proměnných prostředí” na stránce 77.
setjmsenv64	Nastaví proměnné prostředí pro spuštění aplikace IBM MQ classes for JMS v 64bitovém prostředí JVM na systémech UNIX and Linux , jak je popsáno v tématu “Nastavení proměnných prostředí” na stránce 77.

Poznámka:

1. V systému Windowsmá název souboru příponu .bat.

Podpora pro OSGi

OSGi poskytuje rámec, který podporuje implementaci aplikací jako balíky. Devět svazků balíků OSGi je dodáváno jako součást produktu IBM MQ classes for JMS.

Specifikace OSGi poskytuje obecný účel, zabezpečený a spravovaný rámec Java , který podporuje implementaci aplikací, které jsou dodávány ve formě balíků. Zařízení vyhovující specifikaci OSGi mohou stahovat a instalovat balíky a odebírat je, pokud již nejsou zapotřebí. Rámec spravuje instalaci a aktualizaci balíků v dynamickém a přizpůsobitelném stylu.

Produkt IBM MQ classes for JMS obsahuje následující svazky balíků OSGi.

com.ibm.msg.client.osgi.jms*version_number*.jar
: Společná vrstva kódu v produktu IBM MQ classes for JMS. Informace o vrstvené architektuře tříd produktu IBM MQ pro produkt JMSnaleznete v tématu TřídyIBM MQ pro architekturu JMS.

com.ibm.msg.client.osgi.jms.prereq_*version_number*.jar
: Předpokládané soubory JAR (archiv Java) pro obecnou vrstvu.

com.ibm.msg.client.osgi.commonservices.j2se_*version_number*.jar
: Běžné služby pro aplikace Java Platform, Standard Edition (Java SE).

com.ibm.msg.client.osgi.nls_*version_number*.jar
: Zprávy pro obecnou vrstvu.

com.ibm.msg.client.osgi.wmq_*version_number*.jar
Poskytovatel systému zpráv produktu IBM MQ v produktu IBM MQ classes for JMS. Informace o vrstvené architektuře produktu IBM MQ classes for JMSnaleznete v tématu Architektura produktuIBM MQ pro architekturu JMS.

com.ibm.msg.client.osgi.wmq.prereq_*version_number*.jar
Předpokládané soubory JAR pro poskytovatele systému zpráv produktu IBM MQ .

com.ibm.msg.client.osgi.wmq.nls_*version_number*.jar
Zprávy pro poskytovatele systému zpráv produktu IBM MQ .

com.ibm.mq.osgi.allclient_*version_number*.jar
Tento soubor JAR umožňuje aplikacím používat jak IBM MQ classes for JMS , tak i kód IBM MQ classes for Java, a také obsahuje kód pro zpracování zpráv PCF.

com.ibm.mq.osgi.allclientprereqs_*version_number*.jar
Tento soubor JAR poskytuje předpoklady pro `com.ibm.mq.osgi.allclient_`*`version_number`*`.jar` , kde *`version_number`* je číslo verze instalovaného produktu IBM MQ .

Balíky jsou nainstalované do podadresáře `java/lib/OSGi` vaší instalace produktu IBM MQ nebo složky `java\lib\OSGi` v systému Windows.

V produktu IBM MQ 8.0použijte balíky `com.ibm.mq.osgi.allclient_8.0.0.0.jar`a `com.ibm.mq.osgi.allclientprereqs_8.0.0.0.jar` pro všechny nové aplikace. Použijete-li tyto balíky k odstranění omezení, že nebude možné spustit jak produkt IBM MQ classes for JMS , tak produkt IBM MQ classes for Java v rámci stejného rámce OSGi, všechna ostatní omezení se však budou i nadále používat. Pro verze produktu před verzí IBM MQ 8.0platí toto omezení použití produktu IBM MQ classes for JMS nebo IBM MQ classes for Java .

Balík `com.ibm.mq.osgi.java`*`version_number`*`.jar`, který je nainstalován také do podadresáře `java/lib/OSGi` vaší instalace IBM MQ nebo složky `java\lib\OSGi` v systému Windows, je součástí produktu IBM MQ classes for Java. Tento balík se nesmí načíst do běhového prostředí OSGi, které má načtené IBM MQ classes for JMS .

Balíky OSGi pro produkt IBM MQ classes for JMS byly zapsány do specifikace OSGi Release 4. Nepracují v prostředí OSGi Release 3.

Musíte správně nastavit systémovou cestu nebo cestu ke knihovně tak, aby běhové prostředí OSGi mohlo najít všechny požadované soubory DLL nebo sdílené knihovny.

Použijete-li balíky OSGi pro produkt IBM MQ classes for JMS, dočasná témata nebudou fungovat. Kromě toho nejsou třídy uživatelské procedury kanálu zapsané v produktu Java podporovány z důvodu inherentních problémů při načítání tříd v prostředí zavaděče více tříd, jako je například OSGi. Balík uživatelů si může být vědom balíků produktu IBM MQ classes for JMS , ale balíky produktu IBM MQ classes for JMS si nejsou vědomy žádného balíku uživatele. V důsledku toho zavaděč tříd použitý v balíku IBM MQ classes for JMS nemůže načíst třídu ukončení kanálu, která se nachází v uživatelském balíku.

Další informace o specifikaci OSGi naleznete na webu OSGi Alliance .

Získání IBM MQ classes for JMS samostatně

IBM MQ classes for JMS jsou dostupné v samorozbalovacím souboru JAR, který si můžete stáhnout z Fix Central , chcete-li získat pouze soubory JAR produktu IBM MQ classes for JMS , pro implementaci do nástroje pro správu softwaru nebo pro použití se samostatnými klientskými aplikacemi.

Než začnete

Před spuštěním této úlohy se ujistěte, že je na vašem počítači nainstalováno prostředí Java runtime environment (JRE) a že prostředí JRE bylo přidáno do systémové cesty.

Instalační program produktu Java , který se používá v tomto instalačním procesu, nevyžaduje spuštění jako uživatel root nebo žádný specifický uživatel. Jediným požadavkem je, aby uživatel byl spuštěn jako přístup pro zápis do adresáře, do kterého mají být soubory uloženy.

Informace o této úloze

Před IBM MQ 8.0nejsou produkty IBM WebSphere MQ classes for Java nebo IBM WebSphere MQ classes for JMS k dispozici jako samostatné stahování. Pokud v produktu IBM WebSphere MQ 7.5 nebo v dřívějších verzích vyvíjíte a spouštíte jazykové aplikace produktu Java , které používají buď IBM WebSphere MQ classes for Java nebo IBM WebSphere MQ classes for JMS, musíte je nainstalovat buď provedením úplné instalace serveru, nebo instalací jednoho z klientů SupportPacs na systém, kde je aplikace vyvíjována, a na systému, kde bude aplikace spuštěna. Tato instalace nainstaluje mnoho dalších souborů, než soubory IBM WebSphere MQ classes for Java a IBM WebSphere MQ classes for JMS .

Nicméně z produktu IBM MQ 8.0jsou k dispozici následující soubory v samorozbalovacím souboru JAR, který minimalizuje velikost stahování a instalace a dobu potřebnou k provedení instalace:

- :NONE.IBM MQ classes for JMS
- :NONE.IBM MQ classes for Java
- Adaptér prostředků produktu IBM MQ
- Balíky OSGi produktu IBM MQ

Když spustíte spustitelný soubor JAR, zobrazí se licenční smlouva IBM MQ , která musí být přijata. Požádá o adresář, do kterého se mají instalovat produkty IBM MQ classes for Java, IBM MQ classes for JMS, adaptér prostředků a svazky balíků OSGi. Pokud vybraný instalační adresář neexistuje, je vytvořen a jsou instalovány programové soubory. Pokud však adresář existuje, je ohlášena chyba a nejsou nainstalovány žádné soubory.

Postup

1. Stáhněte soubor JAR produktu IBM MQ Java z webu Fix Central.

   Chcete-li vyhledat nejnovější verzi, která je k dispozici ke stažení, zadejte frázi `"Java"` do pole **Textové vyhledávání** . Název souboru, který má být stažen, je ve formátu *V.R.M.F*`-WS-MQ-Install-Java-All.jar` , kde *V.R.M.F* je číslo verze produktu, například `9.0.0.0`.

   Pokud nemůžete soubor najít, ujistěte se, že **Vybraný produkt** je `WebSphere MQ` a **Verze** je `9.0`.
2. Spusťte instalaci z adresáře, do kterého jste stáhli soubor.

   Chcete-li spustit instalaci, zadejte příkaz v následujícím formátu:

   ```
   java -jar V.R.M.F-WS-MQ-Install-Java-All.jar
   ```

   kde *V.R.M.F* je číslo verze produktu, například `9.0.0.0`, a *V.R.M.F*`-WS-MQ-Install-Java-All.jar` je název souboru, který byl stažen z Fix Central.

   Chcete-li například nainstalovat produkt IBM MQ classes for JMS for IBM MQ 9.0.0.0, použijte následující příkaz:

   ```
   java -jar 9.0.0.0-WS-MQ-Install-Java-All.jar
   ```

   Poznámka: Chcete-li provést tuto instalaci, musíte mít na svém počítači nainstalováno prostředí JRE a přidat k systémové cestě.

   Když zadáte příkaz, zobrazí se následující informace:

   ```
   Než budete moci použít, extrahovat nebo instalovat produkt IBM MQ 9.0, musíte přijmout
   podmínky 1. IBM International License Agreement for Evaluation of
   Programy 2. Licenční smlouva IBM International Program License Agreement a další
   informace o licenci. Prosím, přečtěte si pozorně následující licenční smlouvy.

   Licenční smlouva je samostatně zobrazitelná pomocí
   --viewLicenseVolba dohody.

   Chcete-li nyní zobrazit licenční podmínky, stiskněte klávesu Enter. Chcete-li ji přeskočit,
   stiskněte klávesu 'x'.
   ```
3. Přezkoumejte a přijměte licenční podmínky:

   a) Chcete-li zobrazit licenci, stiskněte klávesu Enter.

   Alternativně stisknutím x přeskočíte zobrazení licence.

Po zobrazení licence nebo ihned po výběru x se zobrazí následující zpráva:

```
Další informace o licenci jsou samostatně viditelné pomocí
--viewLicenseInformace o volbě.

Stisknutím klávesy Enter zobrazíte další informace o licenci nyní, nebo 'x' pro přeskočení.
```

b) Chcete-li zobrazit další licenční podmínky, stiskněte klávesu Enter.

Alternativně stisknutím x přeskočíte zobrazení dodatečných licenčních podmínek.

Po zobrazení dalších licenčních podmínek nebo okamžitě, pokud vyberete volbu x, zobrazí se následující zpráva:

```
Vyberete-li níže uvedenou volbu "Souhlasím", souhlasíte s podmínkami
případně licenční smlouvy a podmínky, které nejsouIBM , pokud jsou použitelné. Pokud ne,
Souhlasím, vyberte volbu "Nesouhlasím".

Vyberte [ 1] Souhlasím, nebo [ 2] Nesouhlasím:
```

c) Chcete-li přijmout licenční smlouvu a pokračovat výběrem instalačního adresáře, vyberte volbu 1.

Případně, výběr 2 ukončí instalaci okamžitě.

Vyberete-li 1, zobrazí se následující zpráva:

```
Zadejte adresář pro soubory produktu nebo ponechte pole prázdné, chcete-li přijmout výchozí
hodnotu.
Výchozí cílový adresář je H: \WMQ

Cílový adresář pro soubory produktu?
```

4. Určete instalační adresář pro adaptér prostředků:

- Chcete-li instalovat soubory produktu do výchozího umístění, stiskněte klávesu Enter bez uvedení hodnoty.
- Chcete-li instalovat soubory produktu do jiného umístění z výchozího umístění, zadejte název adresáře, do kterého chcete nainstalovat soubory produktu, a pak stiskněte klávesu Enter, abyste spustili instalaci.

Název adresáře, který uvedete, nesmí již existovat, jinak když spustíte instalaci, je ohlášena chyba a nejsou nainstalovány žádné soubory.

Pokud tento instalační adresář ještě neexistuje, bude vytvořen instalační adresář a v tomto adresáři budou instalovány soubory programů. Během instalace se v instalačním adresáři, který jste vybrali, vytvoří nový adresář s názvem `wmq` . Tři podadresáře, `JavaEE`, `JavaSE`a `OSGi`jsou vytvořeny v adresáři `wmq` s následujícím obsahem:

```
.\JavaEE:
wmq.jmsra.ivt.ear
wmq.jmsra.rar

.\JavaSE:
com.ibm.mq.allclient.jar
com.ibm.mq.traceControl.jar
fscontext.jar
jms.jar
providerutil.jar

.\OSGi:
com.ibm.mq.osgi.allclient_V.R.M.F.jar
com.ibm.mq.osgi.allclientprereqs_V.R.M.F.jar
```

kde *V.R.M.F* je číslo verze, vydání, modifikace a opravné sady.

V 9.0.0.3 V 9.0.5 Před IBM MQ 9.0.0 Fix Pack 3 a IBM MQ 9.0.5jsou soubory, které jsou instalovány v adresáři `JavaSE` , zahrnuty do souboru `JSON4J.jar` . Tento soubor JAR se však nevyžaduje, a proto je odebrán ze souboru *V.R.M.F*`-WS-MQ-Install-Java-All.jar` z IBM MQ 9.0.0 Fix Pack 3 a IBM MQ 9.0.5. Také z IBM MQ 9.0.0 Fix Pack 3 a IBM MQ 9.0.5jsou dvě změny na `com.ibm.mq.allclient.jar file`:

- Odkaz na soubor `JSON4J.jar` se odstraní ze souboru cesty ke třídě v souboru typu manifest pro soubor `com.ibm.mq.allclient.jar` .

- Balík `com.ibm.msg.client.mqlight` již není zahrnut do souboru `com.ibm.mq.allclient.jar`.

Když je instalace dokončena, zobrazí se zpráva s potvrzením, jak je zobrazeno v následujícím příkladu:

```
Extrahování souborů do H: \WMQ \wmq
Všechny soubory produktu se úspěšně extrahovaly.
```

V9.0.0.1 Povolit výpis v IBM MQ classes for JMS

Jako potenciální bezpečnostní riziko bylo identifikováno Java serializace objektů a deserializační mechanizmus. Volba Allowlisting v produktu IBM MQ classes for JMS poskytuje určitou ochranu proti některým rizikům serializace.

Mechanizmus serializace a deserializace objektů produktu Java byl identifikován jako potenciální riziko zabezpečení, protože deserializace vytváří instanci libovolných objektů produktu Java , kde je potenciál pro nekalciózně odeslaná data způsobovat různé problémy. Jedna pozoruhodná aplikace serializace je v produktu Java Message Service (JMS) ObjectMessages , která používá serializaci pro zapouzdření a přenos libovolných objektů.

Serializační povolení je potenciální zmírnění některých rizik, která serializace představuje. Explicitně určením, které třídy lze zapouzdřovat a extrahovat z ObjectMessages, umožňuje povolení některé ochrany proti některým serializačním rizikům.

Povolit výpis v IBM MQ classes for JMS

Viz:

- “Povolit koncepce” na stránce 110 -přehled seznamu povolených
- “Nastavení a použití seznamu povolených JMS” na stránce 113 pro informace o tom, jak nastavit seznam povolených položek
- “Povolit výpis v WebSphere Application Server” na stránce 115 pro informace o tom, jak jste nastavili seznam povolených položek v produktu WebSphere Application Server.

Související pojmy

“Spuštění aplikací produktu IBM MQ classes for JMS v rámci struktury Java Security Manager” na stránce 89

Produkt IBM MQ classes for JMS může být spuštěn se zapnutým správcem zabezpečení produktu Java . Chcete-li úspěšně spustit aplikace se zapnutým serverem Java Security Manager , musíte nakonfigurovat prostředí Java virtual machine (JVM) s vhodným konfiguračním souborem zásad.

V9.0.0.1 *Povolit koncepce*

V produktu IBM MQ classes for JMSposkytuje podpora pro povolení seznamu tříd v implementaci rozhraní JMS ObjectMessage potenciální zmírnění rizik zabezpečení, která se potenciálně týkají serializace objektů Java a deserializovacího mechanismu.

Povolit výpis v IBM MQ classes for JMS

Důležité:

Termín *allowlist* nahradil termín *whitelist*. V případě produktu IBM MQ 9.0 a novějších vydání toto zahrnuje názvy systémových vlastností produktu Java zmíněné v tomto tématu (**`com.ibm.mq.jms.*`**). Nemusíte měnit žádnou existující konfiguraci. Předchozí názvy systémové vlastnosti budou také pracovat.

IBM MQ classes for JMS podporuje povolení seznamu tříd v implementaci rozhraní JMS ObjectMessage .

Seznam allowlist definuje, které třídy produktu Java mohou být serializovány pomocí ObjectMessage.setObject() a deserializovány s objekty ObjectMessage.getObject().

Pokusy o serializaci nebo deserializaci instance třídy, která není zahrnuta v seznamu allowlist s parametrem ObjectMessage , způsobí vygenerování výjimky javax.jms.MessageFormatException a její příčinou je výjimka java.io.InvalidClassException .

Vytvoření seznamu povolených položek

Důležité: IBM MQ classes for JMS nelze distribuovat s allowlist. Výběr tříd, které mají být přeneseny pomocí ObjectMessages , je volba návrhu aplikace a produkt IBM MQ jej nemůže předjímat.

Z tohoto důvodu umožňuje mechanizmus allowlisting pro dva režimy provozu:

Zjišťování
V tomto režimu vytvoří mechanizmus výpis úplných názvů tříd, které hlásí všechny třídy, které byly sledovány, aby byly serializovány nebo deserializovány v ObjectMessages.

Vynucení
V tomto režimu mechanizmus vynucuje povolení, odmítá pokusy o serializaci nebo deserializaci tříd, které nejsou v seznamu povolených položek.

Chcete-li tento mechanismus použít, musíte nejprve spustit příkaz DISCOVERY a shromáždit seznam aktuálně serializovaných a deserializovaných tříd, přezkoumat seznam a použít jej jako základ pro seznam povolených operací. Možná bude vhodné použít seznam nezměněný, ale před tím, než se rozhodnete toto provést, musí být seznam přezkoumán.

Řízení mechanismu allowlisting

K dispozici jsou tři systémové vlastnosti pro ovládání mechanizmu allowlisting:

com.ibm.mq.jms.allowlist
Tato vlastnost může být zadána jedním z následujících způsobů:

- Název cesty k souboru, který obsahuje seznam povolených, ve formátu identifikátoru URI souboru (který je počínaje `file:`). V režimu DISCOVERY je tento soubor zapisován mechanizmem allowlisting. Soubor nesmí existovat. Pokud soubor existuje, mechanizmus vygeneruje výjimku a nepřepisuje jej. V režimu ENFORCEMENT je tento soubor přečten mechanismem allowlisting.
- Čárkami oddělené plně kvalifikované názvy tříd, které tvoří seznam povolených.

Není-li tato vlastnost nastavena na nenastavená vlastnost, mechanizmus allowlist je neaktivní.

Používáte-li prostor Java Security Manager, musíte zajistit, aby soubory JAR produktu IBM MQ classes for JMS měly přístup pro čtení a zápis do tohoto souboru.

com.ibm.mq.jms.allowlist.discover

- Je-li tato vlastnost nenastavena nebo nastavena na hodnotu false, bude mechanizmus allowlist spuštěn v režimu ENFORCEMENT.
- Je-li tato vlastnost nastavena na hodnotu true a jako identifikátor URI souboru byl určen parametr allowlist, bude mechanizmus allowlist spuštěn v režimu DISCOVERY.
- Je-li tato vlastnost nastavena na hodnotu true a byl-li jako seznam názvů tříd zadán seznam povolených názvů, aktivuje mechanizmus allowlist vhodnou výjimku.
- Je-li tato vlastnost nastavena na hodnotu true a seznam allowlist nebyl zadán pomocí vlastnosti com.ibm.mq.jms.allowlist, mechanizmus allowlist je neaktivní.
- Je-li tato vlastnost nastavena na hodnotu true a soubor allowlist již existuje, mechanizmus allowlist vyvolá výjimku java.io.InvalidClassException a položky nebudou přidány do souboru.

com.ibm.mq.jms.allowlist.mode
Tato řetězcová vlastnost může být zadána libovolným ze tří způsobů:

- Je-li tato vlastnost nastavena na SERIALIZE, pak režim ENFORCEMENT provede ověření allowlist pouze v metodě ObjectMessage.setObject().
- Je-li tato vlastnost nastavena na hodnotu DESERIALIZE, pak režim ENFORCEMENT provádí ověřování allowlist pouze v metodě ObjectMessage.getObject().
- Je-li tato vlastnost nenastavena nebo nastavena na jakoukoli jinou hodnotu, pak režim ENFORCEMENT provede ověření povolení na obou metodách ObjectMessage.getObject() a ObjectMessage.setObject().

Formát souboru allowlist

Toto jsou hlavní funkce ve formátu souboru allowlist:

- Soubor allowlist je ve výchozím kódování souboru platformy s čárovým koncovkami odpovídajícího platformům.

  Poznámka: Je-li použit soubor allowlist, je tento soubor vždy zapsán a načten s použitím výchozího kódování souboru pro prostředí JVM.

  To je v pořádku, pokud soubor allowlist je generován kterýkoli z následujících způsobů:

  - z/OS Generováno samostatnou aplikací spuštěnou v produktu z/OS a používaná jinými samostatnými aplikacemi, které jsou rovněž spuštěny v produktu z/OS.
  - Generováno aplikací spuštěnou uvnitř portálu WebSphere Application Server na libovolné platformě a používaná jinou instancí produktu WebSphere Application Server.
  - Multi Generovaná samostatnou aplikací spuštěnou v produktu IBM MQ for Multiplatformsa používaná jinými samostatnými aplikacemi spuštěnými v systému IBM MQ for Multiplatformsnebo aplikacemi spuštěnými v rámci produktu WebSphere Application Server na libovolné platformě.

  Protože však produkt WebSphere Application Server používá kódování ASCII a samostatné prostředí JVM používá EBCDIC, bude při generování souboru allowlist problémy kódování souboru, pokud je soubor allowlist vygenerován některým z následujících způsobů:

  - Generováno na z/OS, pak používané samostatnými aplikacemi spuštěnými na platformě jiné než z/OS nebo WebSphere Application Server.
  - Vygenerováno buď produktem WebSphere Application Server , nebo samostatnou aplikací spuštěnou na jiné platformě než z/OSa poté ji lze použít samostatnou aplikací v systému z/OS.
- Každý není-prázdný řádek obsahuje úplný název třídy. Prázdné řádky se ignorují.
- Komentáře lze zahrnout-cokoli následujícího po znaku '#', na konec řádku, se ignoruje.
- Existuje velmi základní mechanizmus wildsorting:
  - '*' může být **poslední** prvek názvu třídy.
  - Hodnota '*' odpovídá prvku **jediný** názvu třídy, tj. třídě, ale ne součástí balíku.

  Takže `com.ibm.mq.*` by odpovídalo `com.ibm.mq.MQMessage` , ale ne `com.ibm.mq.jmqi.remote.api.RemoteFAP`.

  Použití zástupných znaků nepracuje pro třídy ve výchozím balíku, které jsou určeny pro třídy bez explicitního názvu balíku, takže název třídy "*" je odmítnut.
- Chybně naformátované soubory seznamu povolených souborů, například soubory obsahující položku jako `com.ibm.mq.*.Message`, kde zástupný znak není posledním prvkem, je výsledkem vyvolání výjimky java.lang.IllegalArgumentException .
- Prázdný soubor allowlist má vliv na úplné zablokování použití ObjectMessage.

Formát seznamu allowList jako seznam oddělený čárkami

Stejný mechanizmus wildcars je k dispozici pro seznam povolených položek jako seznam oddělený čárkami.

- Systém '*' může být rozšířen operačním systémem, pokud je zadán na příkazovém řádku nebo ve skriptu shellu nebo dávkovém souboru, takže může vyžadovat speciální zacházení.
- Znak komentáře '#' lze použít pouze v případě, že je zadán soubor. Je-li parametr allowlist zadán jako seznam názvů tříd oddělených čárkami, pak za předpokladu, že operační systém nebo shell jej nezpracuje, protože se jedná o výchozí znak komentáře v mnoha shellech UNIX nebo Linux , je považován za normální znak.

Kdy se stane objekt allowlisting?

Allowlisting je zahájen, když aplikace poprvé spustí metodu ObjectMessage setMessage() nebo getMessage().

Systémové vlastnosti jsou vyhodnoceny, soubor allowlist je otevřen a v režimu ENFORCEMENT, seznam povolených tříd jsou načteny, když je mechanismus inicializován. V tomto okamžiku je položka zapsána do souboru protokolu produktu IBM MQ JMS pro aplikaci.

Když je mechanismus inicializován, jeho parametry nemusí být změněny. Doba inicializace není snadno předpovězena, protože závisí na chování aplikace. Nastavení vlastností systému a obsah souboru allowlist by proto měly být považovány za pevné od okamžiku spuštění aplikace. Neměňte vlastnosti nebo obsah souboru allowlist, když je aplikace spuštěna, protože výsledky nejsou garantovány.

Body k posouzení

Nejlepším přístupem ke zmírnění rizik, která jsou součástí serializace Java , by bylo prozkoumání alternativních přístupů k přenosu dat, jako je například použití JSON namísto ObjectMessage. Pomocí mechanizmů produktu Advanced Message Security (AMS) můžete přidat další zabezpečení tím, že zajistíte, že zprávy pocházejí z důvěryhodných zdrojů.

Použijete-li ke své aplikaci mechanizmus Java Security Manager , musíte udělit následující oprávnění:

- FilePermission na libovolném souboru allowlist, který používáte, s oprávněním ke čtení pro režim ENFORCEMENT, zapište oprávnění pro režim DISCOVER.
- PropertyPermission (čtení) na vlastnostech **com.ibm.mq.jms.allowlist**, **com.ibm.mq.jms.allowlist.discover**a **com.ibm.mq.jms.allowlist.mode** .

Další informace

V9.0.0.1

Další informace o seznamech allowlists najdete v tématech “Nastavení a použití seznamu povolených JMS” na stránce 113 a “Povolit výpis v WebSphere Application Server” na stránce 115 .

Související pojmy

“Spuštění aplikací produktu IBM MQ classes for JMS v rámci struktury Java Security Manager” na stránce 89

Produkt IBM MQ classes for JMS může být spuštěn se zapnutým správcem zabezpečení produktu Java . Chcete-li úspěšně spustit aplikace se zapnutým serverem Java Security Manager , musíte nakonfigurovat prostředí Java virtual machine (JVM) s vhodným konfiguračním souborem zásad.

V9.0.0.1 *Nastavení a použití seznamu povolených JMS*

Tyto informace vám říkají, jak seznam povolených funguje a jak nastavíte jednu z funkcí obsažených v produktu IBM MQ classes for JMS k vygenerování souboru allowlist, který obsahuje seznam typů objektů ObjectMessages , které aplikace může zpracovat.

Než začnete

Důležité:

Termín *allowlist* nahradil termín *whitelist*. V případě produktu IBM MQ 9.0 a novějších vydání toto zahrnuje názvy systémových vlastností produktu Java zmíněné v tomto tématu (**com.ibm.mq.jms.***). Nemusíte měnit žádnou existující konfiguraci. Předchozí názvy systémové vlastnosti budou také pracovat.

Před spuštěním této úlohy se ujistěte, že jste přečetli a porozuměli “Povolit koncepce” na stránce 110

Informace o této úloze

Povolili-li funkci allowlisting, bude produkt IBM MQ classes for JMS používat tuto funkci následujícími způsoby:

- Chce-li aplikace odeslat zprávu ObjectMessage, může ji vytvořit jedním ze dvou způsobů voláním následujících položek:
  - Metoda Session.createObjectMessage(serializovatelnost), předání v objektu, který má být obsažen v rámci zprávy.
  - Metodě Session.createObjectMessage() vytvořte prázdnou hodnotu ObjectMessagea poté volání ObjectMessage.setObject(Serializable) pro uložení objektu, který má být odeslán, do objektu ObjectMessage.

  Je-li volána metoda Session.createObjectMessage(Serializable) nebo ObjectMessage.setObject(Serializable), třídy pro službu JMS kontrolují, zda předaný objekt není typu, který je uveden v seznamu povolených.

  Je-li uveden typ, objekt je serializován a uložen v rámci ObjectMessage. Pokud je však objekt typu, který není v povolenoseznamu, příkaz IBM MQ classes for JMS vygeneruje výjimku JMSException obsahující zprávu:

  ```
  JMSCC0052: Došlo k výjimce při serializaci objektu:
  'java.io.InvalidClassException: < třída objektu >; Třída nemůže být serializována
  nebo deserializovaný, protože nebyl zahrnut v povoleném seznamu '< allowlist>'.
  ```

  zpět do aplikace.

  Důležité: Dojde-li k výjimce ze metody Session.createObjectMessage(Serializable), nebude objekt ObjectMessage vytvořen. Podobně, je-li vyvolána výjimka JMSException z metody ObjectMessage.setObject(Serializable), objekt nebude přidán do ObjectMessage.
- Pokud aplikace přijme zprávu ObjectMessage, volá metodu ObjectMessage.getObject() pro získání objektu obsaženého v tomto objektu. Je-li tato metoda volána, zkontroluje IBM MQ classes for JMS typ objektu obsaženého v ObjectMessage, abyste zjistili, zda je daný objekt typu uvedený v seznamu povolených.

  Pokud ano, objekt je deserializován a vrácen do aplikace. Pokud je však objekt typu, který není v povolenoseznamu, příkaz IBM MQ classes for JMS vygeneruje výjimku JMSException obsahující zprávu:

  ```
  JMSCC0053: Došlo k výjimce při deserializaci zprávy:
  'java.io.InvalidClassException: < třída objektů >; Třída nemusí být
  serializovaný nebo deserializovaný, protože nebyl zahrnut do
  allowlist '< allowlist>'. '
  ```

  zpět do aplikace.

Předpokládejme například, že vaše aplikace obsahuje následující kód k odeslání objektu ObjectMessage obsahujícího objekt typu java.net.URI:

```
java.net.URL testURL = new java.net.URL("https://www.ibm.com/");
ObjectMessage msg = session.createObjectMessage(testURL);
sender.send(msg);
```

Protože není povolen seznam povolení, aplikace je schopna úspěšně vložit zprávu do požadovaného cíle.

Pokud vytvoříte soubor s názvem `C:\allowlist.txt` obsahující jednu položku, `java.net.URL`a spustíte aplikaci znovu se sadou systémových vlastností Java:

```
-Dcom.ibm.mq.jms.allowlist=file:/C:/allowlist.txt
```

funkce allowlist je povolena. Aplikace je stále schopna vytvořit a odeslat ObjectMessage obsahující objekt typu java.net.URI , protože tento typ je uveden v seznamu allowlist.

Pokud však změníte soubor `allowlist.txt` tak, aby soubor obsahoval jedinou položku `java.util.Calendar`, protože funkce allowlist je stále povolena, když volá aplikace:

```
ObjectMessage msg = session.createObjectMessage(testURL);
```

IBM MQ classes for JMS kontroluje seznam povolených položek a zjistil, že neobsahuje položku pro java.net.URI.

V důsledku toho je vyvolána výjimka JMSException obsahující zprávu JMSCC0052 .

Podobně, předpokládejme, že máte jinou aplikaci, která přijímá ObjectMessages pomocí tohoto kódu:

```
ObjectMessage message = (ObjectMessage)receiver.receive(30000);
if (message != null) {
    Object messageBody = objectMessage.getObject();
    if (messageBody instanceof java.net.URI) {
      :    :    :    :    :    :    :
```

Pokud není povolen seznam povolení, aplikace bude moci přijmout ObjectMessages , které obsahují objekt libovolného typu. Aplikace pak zkontroluje, zda je objekt typu java.net.URL před provedením odpovídajícího zpracování.

Pokud nyní spustíte aplikaci se systémovou vlastností Java , postupujte takto:

```
-Dcom.ibm.mq.jms.allowlist=java.net.URL
```

nastavení, funkce allowlisting je zapnuta. Když aplikace volá:

```
Object messageBody = objectMessage.getObject();
```

Metoda ObjectMessage.getObject() vrací pouze objekty typu java.net.URL.

Pokud objekt obsažený v objektu ObjectMessage není tohoto typu, metoda ObjectMessage.getObject() vyvolá výjimku JMSException obsahující zprávu JMSCC0053 . Aplikace pak musí rozhodnout, co se s touto zprávou provést; zpráva může být například přesunuta do fronty nedoručených zpráv pro daného správce front.

Aplikace se vrátí pouze za normálních okolností, pokud je objekt uvnitř ObjectMessage typu java.net.URL.

Postup

1. Spusťte aplikaci, která zpracovává ObjectMessages, s uvedenými vlastnostmi systému Java:

   ```
   -Dcom.ibm.mq.jms.allowlist.discover=true
   -Dcom.ibm.mq.jms.allowlist=file:/<path to your allowlist file>
   ```

   Když se aplikace spustí, vytvoří IBM MQ classes for JMS soubor, který obsahuje typy objektů, které aplikace zpracovala.
2. Po zpracování reprezentativního vzorku objektů ObjectMessages za určité časové období jej zastavte.

   Soubor allowList nyní obsahuje seznam všech typů objektů obsažených v objektu ObjectMessages , které aplikace zpracovávala, zatímco byla spuštěna.

   Pokud jste aplikaci spustili dostatečně dlouho, zahrnuje tento seznam všechny možné typy objektů obsažených v objektu ObjectMessages , které aplikace pravděpodobně zvládne.
3. Restartujte aplikaci s následující sadou systémových vlastností:

   ```
   -Dcom.ibm.mq.jms.allowlist=file:/<path to your allowlist file>
   ```

   To umožní povolit výpis, a pokud IBM MQ classes for JMS zjistí ObjectMessage typu, který není v seznamu povolených, dojde k výjimce JMSException obsahující buď zprávu JMSCC0052 , nebo JMSCC0053 .

V 9.0.0.1 *Povolit výpis v WebSphere Application Server*

Jak použijete IBM MQ classes for JMS allowlisting v WebSphere Application Server.

Důležité:

Termín *allowlist* nahradil termín *whitelist*. V případě produktu IBM MQ 9.0 a novějších vydání toto zahrnuje názvy systémových vlastností produktu Java zmíněné v tomto tématu (**com.ibm.mq.jms.***). Nemusíte měnit žádnou existující konfiguraci. Předchozí názvy systémové vlastnosti budou také pracovat.

Musíte se ujistit, že vaše instalace produktu WebSphere Application Server obsahuje verzi adaptéru prostředků IBM MQ , která podporuje povolení. Tato funkčnost byla přidána do adaptéru prostředků jako součást APAR IT14385.

Další informace o použití těchto dvou produktů najdete v tématu “Společně s IBM MQ a WebSphere Application Server” na stránce 457 .

Jakmile je aplikační server aktualizován, můžete použít systémové vlastnosti produktu Java :

- `-Dcom.ibm.mq.jms.allowlist`
- `-Dcom.ibm.mq.jms.allowlist.discover`

popsáno v části “Nastavení a použití seznamu povolených JMS” na stránce 113.

Poznámka: Musíte nastavit systémové vlastnosti produktu Java jako generické argumenty prostředí JVM, na Java virtual machine použité ke spuštění aplikačního serveru a aplikační server se restartovat, aby se změny projevily.

Další informace naleznete v části *Generické argumenty prostředí JVM* v tématu Nastavení virtuálního počítačeJava .

Chcete-li nastavit vlastnosti, přejděte do okna Java virtual machine v části *Definice procesů* a zadejte příslušný argument.

Následující nastavení:

```
-Dcom.ibm.mq.jms.allowlist=<youruserId>_MyObject
```

způsobí, že aplikační server použije seznam allowlist *youruserId_MyObject*. Aplikační server zpracovává pouze objekty typu.

Následující nastavení:

```
-Dcom.ibm.mq.jms.allowlist.discover=true
-Dcom.ibm.mq.jms.allowlist=file:C/:allowlist.txt
```

nakonfigurujte aplikační server tak, aby používal režim *Discover* , a zaznamenejte podrobnosti o ObjectMessagesproduktu JMS , které aplikační server zpracovává, do souboru `C:\allowlist.txt` .

Následující nastavení:

```
-Dcom.ibm.mq.jms.allowlist=file:C/:allowlist.txt
```

způsobí, že aplikační server načte soubor `C:/allowlist.txt`a použije informace v tomto souboru k určení seznamu allowlist.

Související pojmy

“Spuštění aplikací produktu IBM MQ classes for JMS v rámci struktury Java Security Manager” na stránce 89

Produkt IBM MQ classes for JMS může být spuštěn se zapnutým správcem zabezpečení produktu Java . Chcete-li úspěšně spustit aplikace se zapnutým serverem Java Security Manager , musíte nakonfigurovat prostředí Java virtual machine (JVM) s vhodným konfiguračním souborem zásad.

Konverze řetězcových řetězců v produktu IBM MQ classes for JMS

Produkt IBM MQ classes for JMS používá CharsetEncoders a CharsetDecoders přímo pro převod znakových řetězců. Výchozí chování pro převod znakového řetězce lze nakonfigurovat se dvěma vlastnostmi systému. Zpracování zpráv obsahujících nemapovatelné znaky lze nakonfigurovat prostřednictvím vlastností zprávy pro nastavení akce UnmappableCharactera bajtů náhrady.

Před IBM MQ 8.0byla konverze řetězců v IBM MQ classes for JMS provedena voláním metod `java.nio.charset.Charset.decode(ByteBuffer)` a `Charset.encode(CharBuffer)` .

Použití jedné z těchto metod vede k výchozí náhradě (REPLACE) chybných nebo nepřeložitelných dat. Toto chování může zakrývat chyby v aplikacích a může vést k neočekávaným znakům, například ?, v přeložených datech.

Z produktu IBM MQ 8.0je možné přímo a efektivněji detekovat tyto problémy IBM MQ classes for JMS pomocí CharsetEncoders a CharsetDecoders přímo a explicitně konfigurovat zpracování deformovaných a nepřeložitelných dat. Výchozí chování je takové problémy REPORT , které vyvolává vhodné `MQException`.

Konfigurace

Překládání z UTF-16 (znakové znázornění použité v Java) na nativní znakovou sadu, jako je UTF-8, se označuje jako `encoding`, zatímco překlad v opačném směru se označuje jako `decoding`.

V současné době dekóduje výchozí chování produktu `CharsetDecoders`a hlásí chyby vyvoláním výjimky.

Jedno nastavení se používá k uvedení `java.nio.charset.CodingErrorAction` pro řízení zpracování chyb při kódování i dekódování. Jedno další nastavení se používá k řízení náhradního bajtu, nebo bajtů, při kódování. Výchozí řetězec náhrady Java bude použit při dekódování operací.

UnmappableCharacterNastavení akce a nahrazení v bajtech třídy IBM MQ pro produkt JMS

V produktu IBM MQ 8.0jsou k dispozici následující dvě vlastnosti pro nastavení akce UnmappableCharactera náhradních bajtů. Odpovídající definice konstanty jsou v `com.ibm.msg.client.wmq.WMQConstants`

AKČNÍ AKCE JMS_IBM_UNMAPPABLE_ACTION
Nastaví nebo získává `CodingErrorAction` , aby se použil, když nemůže být znak mapován v operaci kódování nebo dekódování.

Tuto hodnotu byste měli nastavit následujícím způsobem: `CodingErrorAction.{REPLACE|REPORT|IGNORE}.toString()` :

```
public static final String JMS_IBM_UNMAPPABLE_ACTION = "JMS_IBM_Unmappable_Action";
```

NÁHRADA JMS_IBM_UNMAPPABLE_REPLACEMENT
Nastaví nebo získává náhradní bajty, aby bylo možné použít, když nelze namapovat znak v operaci kódování.

Výchozí řetězec náhrady Java se používá při dekódování operací.

```
public static final String JMS_IBM_UNMAPPABLE_REPLACEMENT = "JMS_IBM_Unmappable_Replacement";
```

Vlastnosti `JMS_IBM_UNMAPPABLE_ACTION` a `JMS_IBM_UNMAPPABLE_REPLACEMENT` mohou být nastaveny na místa určení nebo zprávy. Hodnota nastavená ve zprávě přepíše hodnotu nastavenou v cíli, do kterého se zpráva odesílá.

Všimněte si, že produkt `JMS_IBM_UNMAPPABLE_REPLACEMENT` musí být nastaven jako jediný bajt.

Systémové vlastnosti pro nastavení výchozích hodnot systému

V produktu IBM MQ 8.0jsou k dispozici následující dvě vlastnosti systému produktu Java , které slouží ke konfiguraci výchozího chování při převodu znakových řetězců.

com.ibm.mq.cfg.jmqi.UnmappableCharacterAction
Určuje akci, která má být provedena pro nepřeložitelná data při kódování a dekódování. Hodnota může být `REPORT`, `REPLACE`nebo `IGNORE`.

com.ibm.mq.cfg.jmqi.UnmappableCharacterReplacement
Nastaví nebo získává náhradní bajty, aby se aplikoval, když nelze namapovat znak v operaci kódování Výchozí řetězec náhrady Java se používá při dekódování operací.

Chcete-li se vyhnout nejasnostem mezi znakem Java a rodilými bajtovými reprezentacemi, měli byste uvést `com.ibm.mq.cfg.jmqi.UnmappableCharacterReplacement` jako desítkové číslo představující nahrazovací byt v nativní znakové sadě.

Například dekadická hodnota ?jako nativní bajt je 63, pokud je nativní znaková sada založená na ASCII, jako je ISO-8859-1, zatímco je nativní znaková sada 111, je-li nativní znaková sada EBCDIC.

Poznámka: Všimněte si, že pokud má objekt MQMD nebo MQMessage buď sadu polí **`unmappableAction`** , nebo **`unMappableReplacement`** , pak hodnoty těchto polí mají přednost před vlastnostmi systému Java . To umožňuje přepsat hodnoty zadané ve vlastnostech systému Java pro každou zprávu, je-li to nutné.

Související pojmy

"Konverze řetězcových řetězců v produktu IBM MQ classes for Java" na stránce 312
Produkt IBM MQ classes for Java používá CharsetEncoders a CharsetDecoders přímo pro převod znakových řetězců. Výchozí chování pro převod znakového řetězce lze nakonfigurovat se dvěma vlastnostmi systému. Zpracování zpráv, které obsahují nemapovatelné znaky, lze konfigurovat prostřednictvím `com.ibm.mq.MQMD`.

Zápis aplikací IBM MQ classes for JMS

Po stručném úvodu k modelu produktu JMS toto téma obsahuje podrobné pokyny k tomu, jak psát aplikace IBM MQ classes for JMS .

Model produktu JMS

Model produktu JMS definuje sadu rozhraní, která mohou aplikace produktu Java používat k provádění operací systému zpráv. IBM MQ classes for JMS, jako poskytovatel JMS definuje, jak se objekty JMS vztahují k koncepcím IBM MQ . Specifikace JMS očekává určité objekty JMS , které mají být spravovány objekty. Produkt JMS 2.0 zavádí zjednodušené rozhraní API, přičemž zachová také klasické rozhraní API z produktu JMS 1.1.

Specifikace JMS a sada javax.jms definují sadu rozhraní, která mohou aplikace produktu Java používat k provádění operací systému zpráv.

V produktu IBM MQ 8.0produkt podporuje JMS 2.0 verzi standardu JMS, která zavádí zjednodušené rozhraní API a zároveň zachovává klasické rozhraní API z produktu JMS 1.1.

Zjednodušené API

Produkt JMS 2.0 zavádí zjednodušené rozhraní API a zároveň zachovává doménová specifická rozhraní a nezávislá rozhraní domén z produktu JMS 1.1. Zjednodušené rozhraní API snižuje počet objektů potřebných k odesílání a přijímání zpráv a skládá se z následujících rozhraní:

ConnectionFactory
ConnectionFactory je spravovaný objekt, který používá klient produktu JMS k vytvoření připojení. Toto rozhraní se také používá v klasickém rozhraní API.

JMSKontext
Tento objekt kombinuje objekty Connection a Session z klasického rozhraní API. JMSKontextové objekty lze vytvořit z jiných kontextových objektů JMS, přičemž základní připojení bude duplikováno.

JMSProducent
Producent produktu JMSje vytvořen kontextem produktu JMSa používá se k odesílání zpráv do fronty nebo tématu. Objekt Producent produktu JMSzpůsobuje vytvoření objektů, které jsou nezbytné k odeslání zprávy.

JMSSpotřebitel
Odběratel produktu JMSje vytvořen kontextem produktu JMSa slouží k příjmu zpráv z tématu nebo z fronty.

Zjednodušené rozhraní API má několik efektů:

- Kontextové objekty produktu JMSvždy automaticky spustí základní připojení.

- JMSProducenti a JMSSpotřebitelé mohou nyní pracovat přímo se zprávami, aniž by museli získat celý objekt zprávy, a to pomocí metody Message's getBody .
- Vlastnosti zprávy lze nastavit na objektu Producent produktu JMSpomocí řetězení metod, před odesláním 'těla', obsahu zpráv. Producent produktu JMSbude zpracovávat vytváření všech objektů, které jsou potřebné k odeslání zprávy. Pomocí produktu JMS 2.0lze nastavit vlastnosti a odeslat zprávu následujícím způsobem:

```
context.createProducer().
setProperty("foo", "bar").
setTimeToLive(10000).
setDeliveryMode(NON_PERSISTENT).
setDisableMessageTimestamp(true).
send(dataQueue, body);
```

Produkt JMS 2.0 také zavádí sdílené odběry, ve kterých lze sdílet zprávy mezi více spotřebiteli. Všechny odběry JMS 1.1 jsou považovány za nesdílené odběry.

Klasické API

Následující seznam shrnuje hlavní rozhraní API JMS klasického rozhraní API:

Místo určení
Místo určení je místo, kam aplikace odesílá zprávy, nebo je to zdroj, ze kterého aplikace přijímá zprávy, nebo obojí.

ConnectionFactory
Objekt ConnectionFactory zapouzdřuje sadu vlastností konfigurace pro připojení. Aplikace používá továrnu připojení k vytvoření připojení.

Připojení
Objekt připojení zapouzdřuje aktivní připojení aplikace k serveru systému zpráv. Aplikace používá připojení k vytvoření relací.

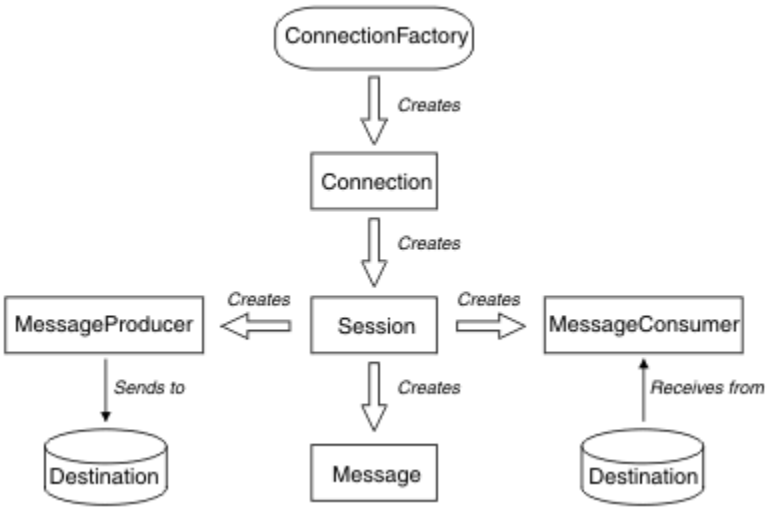
Relace
Relace je jednovláknový kontext pro odesílání a příjem zpráv. Aplikace používá relaci k vytváření zpráv, producentů zpráv a spotřebitelů zpráv. Relace je buď zpracovávána, nebo se nejedná o transakci.

Zpráva
Objekt Message zapouzdřuje zprávu, kterou aplikace odesílá nebo přijímá.

MessageProducer
Aplikace používá producenta zpráv k odesílání zpráv do místa určení.

MessageConsumer
Aplikace používá spotřebitele zpráv k přijetí zpráv odeslaných do místa určení.

Produkt Obrázek 9 na stránce 120 zobrazuje tyto objekty a jejich vztahy.



Obrázek 9. Objekty produktu JMS a jejich vztahy

Místo určení, ConnectionFactorynebo Objekt připojení mohou být použity souběžně různými podprocesy aplikace s více podprocesy, ale objekt Session, MessageProducernebo MessageConsumer nemůže být použit souběžně různými podprocesy. Nejjednodušším způsobem, jak zajistit, aby relace MessageProducernebo MessageConsumer nebyla použita souběžně, je vytvoření samostatného objektu relace pro každý podproces.

Produkt JMS podporuje dva styly systému zpráv:

- Dvoubodový systém zpráv
- Publikování/odběr zpráv

Tyto styly systému zpráv jsou označovány také jako *domény systému zpráv*a v aplikaci můžete kombinovat oba styly systému zpráv. V doméně dvoubodového spojení je cílem fronta a v doméně publikování/odběru je cílem téma.

S verzí produktu JMS před JMS 1.1, programování pro dvoubodovou doménu používá jednu sadu rozhraní a metod, a programování pro doménu publikování/odběru používá jinou sadu. Tyto dvě sady jsou podobné, ale oddělené. V produktu JMS 1.1můžete používat společnou sadu rozhraní a metod, které podporují obě domény systému zpráv. Společná rozhraní poskytují nezávislý pohled domény pro každou doménu systému zpráv. Příkaz Tabulka 14 na stránce 120 vypisuje nezávislá rozhraní domény JMS a jejich odpovídající rozhraní specifická pro doménu.

Tabulka 14. Doména JMS nezávislá na doméně a specifická rozhraní domény

Nezávislá rozhraní domény	**Doména specifická pro doménu pro dvoubodovou doménu**	**Rozhraní specifická pro doménu pro doménu publikování/ odběru**
ConnectionFactory	Továrna QueueConnection	Továrna TopicConnection
Připojení	QueueConnection	TopicConnection
Místo určení	Fronta	Téma
Relace	QueueSession	TopicSession
MessageProducer	QueueSender	TopicPublisher
MessageConsumer	QueueReceiver QueueBrowser	TopicSubscriber

Produkt JMS 2.0 uchovává všechna rozhraní specifická pro doménu a existující aplikace tak mohou tato rozhraní používat i nadále. Pro nové aplikace však zvažte použití nezávislých rozhraní domény produktu JMS 1.1 nebo zjednodušeného rozhraní API produktu JMS 2.0.

V produktu IBM MQ classes for JMSse objekty JMS vztahují k koncepcím produktu IBM MQ následujícími způsoby:

- Objekt připojení má vlastnosti, které jsou odvozeny od vlastností továrny připojení, která byla použita k vytvoření připojení. Tyto vlastnosti řídí způsob, jakým se aplikace připojuje ke správci front. Příklady těchto vlastností jsou název správce front a pro aplikaci, která se připojuje ke správci front v režimu klienta, název hostitele nebo adresa IP systému, v němž je spuštěn správce front.
- Objekt relace zapouzdřuje popisovač připojení IBM MQ , který proto definuje transakční rozsah relace.
- Objekt MessageProducer a objekt MessageConsumer obsahují každý zapouzdřující popisovač objektu IBM MQ .

Při použití IBM MQ classes for JMSse použijí všechna běžná pravidla produktu IBM MQ . Všimněte si zejména, že aplikace může odeslat zprávu do vzdálené fronty, ale může přijmout zprávu pouze z fronty, kterou vlastní správce front, ke kterému je aplikace připojena.

Specifikace JMS očekává objekty ConnectionFactory a Destination Object, aby byly spravovány objekty. Administrátor vytváří a udržuje spravované objekty v centrálním úložišti a aplikace JMS načítá tyto objekty pomocí rozhraní JNDI (Java Naming and Directory Interface).

V produktu IBM MQ classes for JMSje implementace rozhraní Destination abstraktní nadtřída fronty a tématu, takže instance cíle je buď objekt fronty, nebo objekt Topic. Nezávislá rozhraní domény zpracovávají frontu nebo téma jako cíl. Doména systému zpráv pro objekt MessageProducer nebo MessageConsumer je určena podle toho, zda je cílem fronta nebo téma.

V produktu IBM MQ classes for JMS mohou být objekty následujících typů administrované objekty:

- ConnectionFactory
- Továrna QueueConnection
- Továrna TopicConnection
- Fronta
- Téma
- XAConnectionFactory
- Továrna XAQueueConnection
- Továrna XATopicConnection-továrna

Související pojmy

“Použití funkčnosti produktu JMS 2.0” na stránce 289
Produkt JMS 2.0 zavádí několik nových oblastí funkčnosti produktu IBM MQ classes for JMS.

Související informace

Rozhraní jazyka produktu IBM MQ Java

Zprávy produktu JMS

Zprávy produktu JMS se skládají ze záhlaví, vlastností a těla. JMS definuje pět typů těla zprávy.

Zprávy produktu JMS se skládají z následujících částí:

Header
Všechny zprávy podporují stejnou sadu polí záhlaví. Pole záhlaví obsahují hodnoty, které používají klienti i poskytovatelé k identifikaci a směrování zpráv.

Vlastnosti
Každá zpráva obsahuje vestavěné zařízení pro podporu hodnot vlastností definovaných aplikací. Vlastnosti poskytují účinný mechanizmus pro filtrování zpráv definovaných aplikací.

Tělo
JMS definuje pět typů těla zprávy, které pokrývají většinu stylů systému zpráv, které se aktuálně používají:

Proud
Proud primitivních hodnot Java . Je vyplněno a postupně čte.

Mapa
Sada dvojic název-hodnota, kde názvy jsou řetězce a hodnoty, jsou primitivní typy produktu Java . K položkám lze přistupovat sekvenčně nebo náhodně podle názvu. Pořadí položek není definováno.

Text
Zpráva obsahující java.lang.String.

Objekt
Zpráva, která obsahuje serializovatelný objekt Java

Bajtů
Proud neinterpretovaného bajtů. Tento typ zprávy je určen pro doslovně kódování textu zprávy tak, aby odpovídal stávajícímu formátu zprávy.

Pole záhlaví JMSCorrelationID se používá k propojení jedné zprávy s jinou. Obvykle propojí zprávu s odpovědí se svou žádající zprávou. JMSCorrelationID může obsahovat ID zprávy specifické pro poskytovatele, řetězec specifický pro aplikaci nebo hodnotu poskytovatele-nativní bajt [] poskytovatele.

Selektory zpráv v JMS
Zprávy mohou obsahovat hodnoty vlastností definované aplikací. Aplikace může používat selektory zpráv k tomu, aby měly zprávy filtru poskytovatele produktu JMS .

Zpráva obsahuje vestavěnou funkci pro podporu hodnot vlastností definovaných aplikací. V důsledku toho poskytuje mechanismus pro přidání polí záhlaví specifického pro aplikaci do zprávy. Vlastnosti umožňují aplikaci používat selektory zpráv k výběru nebo filtrování zpráv poskytovatele JMS pomocí specifických kritérií aplikace. Vlastnosti definované aplikací musí dodržovat následující pravidla:

- Názvy vlastností musí dodržovat pravidla pro identifikátor selektoru zpráv.
- Hodnoty vlastností mohou být Boolean, byte, short, int, long, float, double a String.
- Předpony názvů JMSX a JMS_ jsou rezervovány.

Hodnoty vlastností jsou nastaveny před odesláním zprávy. Když klient obdrží zprávu, vlastnosti zprávy jsou jen pro čtení. Pokusí-li se klient o nastavení vlastností v tomto bodě, je vyvolána výjimka MessageNotWriteableException . Je-li volána funkce clearProperties , vlastnosti lze nyní číst i zapisovat do pole.

Hodnota vlastnosti může kopírovat hodnotu v těle zprávy. JMS nedefinuje zásadu pro to, co by se dalo provést do vlastnosti. Vývojáři aplikací si však musí být vědomi toho, že poskytovatelé JMS pravděpodobně zpracovávají data v těle zprávy efektivněji než data ve vlastnostech zprávy. Chcete-li dosáhnout nejlepšího výkonu, aplikace musí používat vlastnosti zprávy pouze v případě, že potřebují upravit záhlaví zprávy. Primárním důvodem pro provedení této činnosti je podpora upraveného výběru zpráv.

Selektor zpráv produktu JMS umožňuje klientovi zadávat zprávy, o které se zajímá, a to pomocí záhlaví zprávy. Dodány budou pouze zprávy se záhlavími, které odpovídají selektoru.

Selektory zpráv nemohou odkazovat na hodnoty těla zprávy.

Selektor zpráv odpovídá zprávě, je-li selektor vyhodnocen na hodnotu true, když jsou hodnoty pole záhlaví zprávy a hodnoty vlastností nahrazeny pro odpovídající identifikátory v selektoru.

Selektor zpráv je řetězec se syntaxí, která je založena na dílčí sadě syntaxe podmíněného výrazu SQL92 . Pořadí, ve kterém je vyhodnocován selektor zpráv, je zleva doprava v rámci úrovně priority. Chcete-li toto pořadí změnit, můžete použít závorky. Literály předdefinovaných selektorů a názvy operátorů jsou zapsány zde velkými písmeny; avšak nerozlišují se malá a velká písmena.

Obsah selektoru zpráv

Selektor zpráv může obsahovat:

- literály
  - Řetězcový literál je uzavřený v uvozovkách. Zdvojená uvozovka představuje uvozovku. Příklady jsou 'literal' a 'literal''s '. Jako řetězcové literály Java používají kódování znaků Unicode kódování znaků Unicode.
  - Přesný číselný literál je číselná hodnota bez desetinné čárky, jako například 57, -957 a +62. Čísla v rozsahu Java long jsou podporovány.
  - Přibližný numerický literál je číselná hodnota v exponenciální notaci, například 7E3 nebo -57.9E2, nebo číselná hodnota s desítkovým číslem, například 7., -95.7nebo +6.2. Čísla v rozsahu Java double jsou podporována.
  - Booleovské literály TRUE a FALSE.
- Identifikátory:
  - Identifikátor je neomezená posloupnost znaků Java a Java číslic, přičemž první z nich musí být písmeno Java . Písmeno je libovolný znak, pro který vrací metoda Character.isJava. Písmeno vrátí hodnotu true. To zahrnuje _ a $. Písmeno nebo číslice jsou libovolné znaky, pro které je metoda Character.isJavaLetterOrDigit vrácena true.
  - Identifikátory nemohou být názvy NULL, TRUE nebo FALSE.
  - Identifikátory nemohou být NOT, AND, OR, BETWEEN, LIKE, IN, nebo IS.
  - Identifikátory jsou buď odkazy na pole záhlaví, nebo odkazy na vlastnosti.
  - Identifikátory rozlišují velikost písmen.
  - Odkazy na pole záhlaví zprávy jsou omezeny na:
    - JMSDeliveryMode
    - JMSPriority.
    - JMSMessageID
    - JMSTimestamp
    - JMSCorrelationID
    - JMSType.

    Hodnoty JMSMessageID, JMSTimstamp, JMSCorrelationIDa JMSType mohou mít hodnotu null, a pokud ano, jsou považovány za hodnotu NULL.
  - Všechny názvy začínající řetězcem JMSX jsou názvy vlastností definované produktem JMS.
  - Všechny názvy začínající řetězcem JMS_ jsou název vlastnosti specifický pro poskytovatele.
  - Jakýkoli název, který nezačíná na JMS , je název vlastnosti specifický pro aplikaci. Existuje-li odkaz na vlastnost, která ve zprávě neexistuje, její hodnota je NULL. Pokud existuje, jeho hodnota je odpovídající hodnota vlastnosti.
- Mezera je stejná, jako je definovaná pro Java: mezera, vodorovná karta, posuv na novou stránku a ukončovač řádku.
- Výrazy:
  - Selektor je podmíněný výraz. Selektor, který se vyhodnocuje na skutečné shody; selektor, který se vyhodnocuje na hodnotu false nebo neznámý, se neshoduje.
  - Aritmetické výrazy se skládají z sebe, aritmetických operací, identifikátorů (s hodnotou, se kterou se zachází jako s číselným literálem) a numerickými literály.
  - Podmíněné výrazy se skládají z vlastních, porovnávacích operací a logických operací.
- Standardní bracketing () pro nastavení pořadí, ve kterém jsou výrazy vyhodnocovány, je podporován.
- Logické operátory v pořadí priority: NOT, AND, OR.
- Operátory porovnání: =, >, > =, <, < =, < > (nerovná se).

  - Porovnávají se pouze hodnoty stejného typu. Jedna výjimka je, že je platná pro porovnání přesných číselných hodnot a přibližných číselných hodnot. (Požadovaná konverze typu je definována pomocí pravidel Java číselného povýšení.) Pokud dojde k pokusu o porovnání různých typů, je selektor vždy nepravdivý.
  - Porovnání řetězců a logických hodnot je omezeno na hodnoty = a < >. Dva řetězce jsou shodné pouze v případě, že obsahují stejnou posloupnost znaků.
- Aritmetické operátory v pořadí priority:
  - +,-unární.
  - *,/, násobení a dělení.
  - +,-, sčítání a odečítání.
  - Aritmetické operace na hodnotě NULL nejsou podporovány. Pokud se o tyto pokusy pokusí, úplný selektor je vždy nepravdivý.
  - Aritmetické operace musí používat Java číselnou propagaci.
- arithmetic-expr1 [NOT] BETWEEN arithmetic-expr2 a arithmetic-expr3 operátor porovnání:
  - Věk BETWEEN 15 a 19 je ekvivalentní věku > = 15 AND věk < = 19.
  - Věk NEBUDE BETWEEN 15 a 19 je ekvivalentní věku < 15 OR věk > 19.
  - Má-li některý z výrazů operace BETWEEN hodnotu NULL, je hodnota operace false. Má-li některý z výrazů operace NOT BETWEEN hodnotu NULL, je hodnota operace true.
- identifier [NOT] IN (string-literal1, string-literal2, ...) comparison operator where identifier has a String or NULL value.
  - Země IN ("UK", "US", "Francie") platí pro "Spojené království" a "Peru". Je to ekvivalentní výrazu (Country = 'UK ') OR (Country = 'US') OR (Country = 'France ').
  - Země NOT IN ("UK", "US", "Francie") je nepravdivá pro "Spojené království" a platí pro "Peru". Je ekvivalentní výrazu NOT ((Country = 'UK ') OR (Country = 'US') OR (Country = 'France ')).
  - Je-li identifikátor operace IN nebo NOT IN NULL, hodnota operace je neznámá.
- identifier [NOT] LIKE vzor-hodnota [ESCAPE escape-znak] porovnávací operátor, kde identifikátor má hodnotu řetězce. pattern-value is a string literal, where _ stands for any single character and% stands for any sequence of characters (including the empty sequence). Všechny ostatní znaky jsou stojan pro sebe. Volitelný řídicí znak je jednoznakový řetězcový literál, se znakem, který se používá k úniku speciálního významu _ a% ve vzoru-hodnota.
  - telefon LIKE '12%3' je true pro 123 a 12993 a false pro 1234.
  - slovo LIKE 'l_se' je true pro "prohrát" a false pro "loose".
  - podrýhovaný LIKE '_ %' ESCAPE' \ ' je true pro "_foo" a false pro "bar".
  - telefon NOT LIKE '12%3' je false pro 123 a 12993 a true pro 1234.
  - Je-li identifikátor operace LIKE nebo NOT LIKE NULL, hodnota operace je neznámá.
- Identifikátor IS NULL testuje operátor porovnání pro hodnotu pole záhlaví null nebo chybějící hodnotu vlastnosti.
  - prop_name IS NULL.
- Identifikátor testování operátoru IS NOT NULL testuje existenci hodnoty pole záhlaví bez hodnoty null nebo hodnoty vlastnosti.
  - prop_name IS NOT NULL.

Příklad selektoru zpráv

Následující selektor zpráv vybírá zprávy s typem zprávy auto, barvou modré barvy a váhou větší než 2500 lbs:

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

Hodnoty vlastností NULL

Jak je uvedeno v předchozím seznamu, hodnoty vlastností mohou mít hodnotu NULL. Vyhodnocení výrazů selektoru, které obsahují hodnoty NULL, je definováno sémantikou SQL 92 NULL. Následující seznam uvádí stručný popis těchto sémantik:

- SQL považuje hodnotu NULL za neznámou.
- Porovnání nebo aritmetika s neznámou hodnotou vždy poskytne neznámou hodnotu.
- Operátor IS NULL převede neznámou hodnotu na hodnotu TRUE.
- Operátor IS NOT NULL převede neznámou hodnotu na hodnotu FALSE.

Speciální chování JMSMessageID a JMSCorrelationID

Třídy IBM MQ pro rozhraní JMS obsahují optimalizaci při výběru zpráv z fronty založené na hodnotě JMSMessageID nebo JMSCorrelationID.

Určuje-li aplikace selektor formuláře, postupujte takto:

JMSMessageID= 'ID:*message_id*'

, kde *ID_zprávy* je řetězec obsahující standardní identifikátor zprávy IBM MQ , pak třídy IBM MQ pro JMS používají **MatchOption** MQMO_MATCH_MSG_ID k získání zprávy s uvedeným identifikátorem zprávy.

Chcete-li například získat zprávu s identifikátorem zprávy 414D51207061756C745639314C545320C57C1A5F25ECE602 z fronty, aplikace by měla použít následující selektor zpráv:

```
JMSMessageID='ID:414D51207061756C745639314C545320C57C1A5F25ECE602'
```

Podobně, pokud aplikace uvádí selektor, který má formát:

JMSCorrelationID = 'ID:*correrelation_id*'

, kde *correrelation_id* je řetězec obsahující standardní korelační identifikátor IBM MQ , třídy IBM MQ pro JMS používají **MatchOption** MQMO_MATCH_CORREL_ID k získání zprávy s uvedeným identifikátorem korelace z fronty.

V následujícím příkladu se selektor zpráv používá k získání zprávy, která má korelační identifikátor 414D51207061756C745639314C545320846E5B5F25B1CC02:

```
JMSCorrelationID='ID:414D51207061756C745639314C545320846E5B5F25B1CC02'
```

Pokud selektor zpráv obsahuje hodnotu všech nul buď pro *message_id* , nebo *correlation_id*, pak se shoduje s každou zprávou ve frontě. Pokud například aplikace používá selektor, postupujte takto:

```
JMSMessageID='ID:000000000000000000000000000000000000000000000000'
```

pak se každá zpráva ve frontě považuje za shodu a vrátí se do aplikace.

Další informace o příkazu MQMO_MATCH_MSG_ID a MQMO_MATCH_CORREL_ID **MatchOptions**naleznete v tématu MatchOptions (MQLONG).

Omezení

Ačkoli SQL podporuje pevné desetinné porovnání a aritmetiku, selektory zpráv produktu JMS nikoli. To je důvod, proč jsou přesné číselné literály omezené na ty, které nemají desetinné číslo. Je také důvod, proč existují číselné hodnoty s desetinnou hodnotou jako alternativní znázornění přibližné numerické hodnoty.

Komentáře SQL nejsou podporovány.

Mapování zpráv produktu JMS na zprávy produktu IBM MQ
Zprávy produktu IBM MQ se skládají z deskriptoru zpráv, volitelného záhlaví MQRFH2 a textu zprávy. Obsah zprávy produktu JMS je částečně mapován a částečně zkopírován do zprávy IBM MQ .

Toto téma popisuje, jak je struktura zpráv produktu JMS popsaná v první části této části mapována na zprávu IBM MQ . Je předmětem zájmu programátorů, kteří chtějí předávat zprávy mezi JMS a tradičními aplikacemi IBM MQ . Je také zajímavé pro lidi, kteří chtějí manipulovat se zprávami přenášenými mezi dvěma aplikacemi produktu JMS , například v implementaci produktu IBM Integration Bus .

Tato sekce se nepoužije, pokud aplikace používá připojení v reálném čase ke zprostředkovateli. Když aplikace používá připojení v reálném čase, veškerá komunikace se provádí přímo přes TCP/IP; žádné fronty nebo zprávy IBM MQ se nepodílejí.

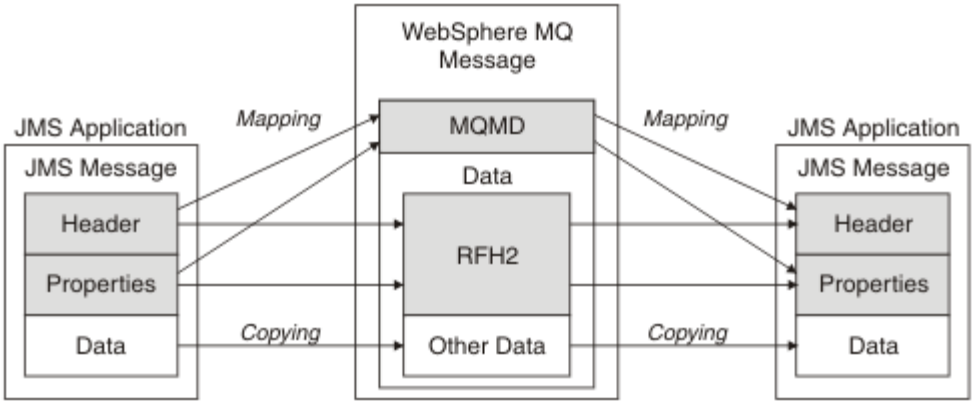
Zprávy produktu IBM MQ se skládají ze tří komponent:

- Deskriptor zpráv produktu IBM MQ (MQMD)
- Záhlaví IBM MQ MQRFH2 .
- Tělo zprávy.

Hodnota MQRFH2 je volitelná a její zahrnutí do odchozí zprávy se řídí příznakem TARGCLIENT ve třídě cíle JMS . Tento parametr můžete nastavit pomocí nástroje pro administraci produktu IBM MQ JMS . Protože MQRFH2 přenáší informace specifické pro produkt JMS, vždy ji zahrňte do zprávy, když odesilatel ví, že přijímající místo určení je aplikace JMS . Obvykle vynechává MQRFH2 při odesílání zprávy přímo do aplikace, která není typuJMS . Důvodem je skutečnost, že taková aplikace neočekává MQRFH2 ve své zprávě IBM MQ .

Pokud příchozí zpráva nemá záhlaví MQRFH2 , objekt Queue nebo Topic odvozený z pole záhlaví JMSReplyTo zprávy má při výchozím nastavení tento příznak nastaven tak, aby zpráva odpovědi odesílaná do fronty nebo tématu také neodeslala záhlaví MQRFH2 . Toto chování včetně záhlaví MQRFH2 ve zprávě odpovědi můžete vypnout pouze v případě, že původní zpráva má záhlaví MQRFH2 , a to nastavením vlastnosti TARGCLIENTMATCHING továrny na připojení na NO.

Obrázek 10 na stránce 126 ukazuje, jak je struktura zprávy JMS transformována na zprávu IBM MQ a znovu opakuji:



Obrázek 10. Způsob transformace zpráv mezi JMS a IBM MQ pomocí záhlaví MQRFH2

Struktury jsou transformovány dvěma způsoby:

Mapování
Pokud deskriptor MQMD obsahuje pole, které je ekvivalentní poli JMS , je pole JMS mapováno na pole MQMD. Další pole MQMD jsou odkryty jako vlastnosti produktu JMS , protože aplikace JMS může potřebovat tato pole při komunikaci s aplikací jinou nežJMS načíst nebo nastavit.

Kopírování
Pokud neexistuje ekvivalent MQMD, je záhlaví nebo vlastnost záhlaví produktu JMS předáno, případně transformováno, jako pole v rámci MQRFH2.

Záhlaví MQRFH2 a JMS
Tato kolekce témat popisuje záhlaví MQRFH verze 2, které obsahuje data specifická pro produkt JMS, která jsou přidružena k obsahu zprávy. Záhlaví MQRFH verze 2 je rozšiřitelné a může také obsahovat další informace, které nejsou přímo přidruženy k produktu JMS. Tento oddíl se však vztahuje pouze na jeho použití produktem JMS.Úplný popis viz MQRFH2 -Pravidla a formátovací záhlaví 2.

Existují dvě části záhlaví, pevná část a část proměnné.

Pevná část
Pevná část je modelovaná na *standardním* IBM MQ vzoru záhlaví a skládá se z následujících polí:

StrucId (MQCHAR4)
Identifikátor struktury.

Musí být MQRFH_STRUC_ID (hodnota: "RFH ") (počáteční hodnota).

MQRFH_STRUC_ID_ARRAY (hodnota: "R", "F", "H", " ") je také definován.

Verze (MQLONG)
Číslo verze struktury.

Musí být MQRFH_VERSION_2 (hodnota: 2) (počáteční hodnota).

StrucLength (MQLONG)
Celková délka MQRFH2, včetně datových polí NameValue.

Hodnota nastavená na StrucLength musí být násobkem 4 (data v polích dat NameValuemohou být doplněna mezerami znaků k dosažení tohoto).

Kódování (MQLONG)
Kódování dat.

Kódování jakýchkoli numerických dat v části zprávy za MQRFH2 (další záhlaví nebo data zprávy za tímto záhlavím).

CodedCharSetId (MQLONG)
Identifikátor kódované znakové sady.

Zastupování libovolných znakových dat v části zprávy za MQRFH2 (další záhlaví nebo data zprávy za tímto záhlavím).

Formát (MQCHAR8)
Název formátu.

Název formátu pro část zprávy následující za MQRFH2.

Příznaky (MQLONG)
Příznaky.

MQRFH_NO_FLAGS = 0. Nejsou nastaveny žádné příznaky.

NameValueCCSID (MQLONG)
Identifikátor kódové sady znaků (CCSID) pro řetězce znaků dat NameValueobsažené v tomto záhlaví. Hodnota NameValueData může být kódována ve znakové sadě, která se liší od ostatních znakových řetězců obsažených v záhlaví (StrucID a Format).

Je-li CCSID NameValuedvoubajtový Unicode CCSID (1200, 13488 nebo 17584), je pořadí bajtů Unicode stejné jako pořadí bajtů numerických polí v MQRFH2. (Například, verze, StrucLengtha NameValueCCSID samotné.)

CCSID NameValuebere hodnoty z následující tabulky:

V 9.0.0

Tabulka 15. Možné hodnoty pro pole NameValueCCSID

CCSID	Význam
1200	UTF-16, nejnovější podporovaná verze Unicode

Tabulka 15. Možné hodnoty pro pole NameValueCCSID (pokračování)	
CCSID	**Význam**
13488	UTF-16, verze Unicode, podmnožina 2.0
17584	UTF-16, verze Unicode 3.0 dílčí sada (obsahuje symbol Euro)
1208	UTF-8, nejnovější podporovaná verze Unicode

Proměnná část

Proměnná část následuje pevnou část. Proměnná část obsahuje proměnný počet složek MQRFH2 . Každá složka obsahuje proměnný počet prvků nebo vlastností. Vlastnosti související se skupinou složek. Hlavičky MQRFH2 vytvořené produktem JMS mohou obsahovat libovolné z následujících složek:

Složka `mcd`

`mcd` obsahuje vlastnosti, které popisují formát zprávy. Například vlastnost `Msd` domény služby zpráv identifikuje zprávu jako typu `JMSTextMessage`, `JMSBytesMessage`, `JMSStreamMessage`, `JMSMapMessage`, `JMSObjectMessage` nebo null.

Složka `mcd` je vždy přítomna ve zprávě JMS obsahující `MQRFH2`.

Vždy se vyskytuje ve zprávě obsahující `MQRFH2` odeslané z IBM Integration Bus. Popisuje doménu, formát, typ a sadu zpráv příslušné zprávy.

Tabulka 16. Název, synonymum, datový typ a složka vlastnosti `mcd`			
Synonymum vlastnosti	**Název vlastnosti**	**Datový typ**	**Složka**
	`mcd.Msd`	`string`	`<mcd><Msd>`*`messageDomain`*`</Msd></mcd>`
	`mcd.Set`	`string`	`<mcd><Set>`*`messageDomain`*`</Set></mcd>`
	`mcd.Type`	`string`	`<mcd><Type>`*`messageDomain`*`</Type></mcd>`
	`mcd.Fmt`	`string`	`<mcd><Fmt>`*`messageDomain`*`</Fmt></mcd>`

Do složky `mcd` nepřidávejte své vlastní vlastnosti.

Složka `jms`

Složka `jms` obsahuje pole záhlaví služby JMS a vlastnosti JMSX, které nelze zcela vyjádřit v `MQMD`. Složka `jms` je vždy přítomna v JMS `MQRFH2`.

Složka `usr`

Složka `usr` obsahuje vlastnosti služby JMS definované aplikací, které jsou přidružené ke zprávě. Složka `usr` je přítomna pouze v případě, že aplikace nastavila vlastnost definovanou aplikací.

Složka `mqext`

`mqext` obsahuje následující typ vlastnosti:

- Vlastnosti, které používá pouze WebSphere Application Server.
- Vlastnosti související se zpožděným doručováním zpráv.

Složka je přítomna, pokud má aplikace nastavenu minimálně jednu definovanou vlastnost IBM nebo používá prodlevu doručení.

Tabulka 17. Název, synonymum, datový typ a složka vlastnosti mqext			
Synonymum vlastnosti	**Název vlastnosti**	**Datový typ**	**Složka**
JMSArmCorrelator	mqext.Arm	string	<mqext><Arm>*armCorrelator*</Arm></mqext>
JMSRMCorrelator	mqext.Wrm	string	<mqext><Wrm>*wrmCorrelator*</Wrm></mqext>
JMSDeliveryTime	mqext.Dlt	i8	<mqext><Dlt>*DeliveryTime*</Dlt></mqext>
JMSDeliveryDelay	mqext.Dly	i8	<mqext><Dly>*DeliveryTime*</Dly></mqext>

Do složky mqext nepřidávejte své vlastní vlastnosti.

Složka mqps

Produkt mqps obsahuje vlastnosti, které jsou používány pouze v rámci publikování/odběru produktu IBM MQ. Tato složka je přítomna pouze, když má aplikace nastavenu minimálně jednu z integrovaných vlastností publikování/odběru.

Tabulka 18. Název, synonymum, datový typ a složka vlastnosti mqps			
Synonymum vlastnosti	**Název vlastnosti**	**Datový typ**	**Složka**
MQTopicString	mqps.Top	string	<mqps><Top>*topicString*</Top></mqps>
MQSubUserData	mqps.Sud	string	<mqps><Sud>*subscriberUserData*</Sud></mqps>
MQIsRetained	mqps.Ret	boolean	<mqps><Ret>*isRetained*</Ret></mqps>
MQPubOptions	mqps.Pub	i8	<mqps><Pub>*publicationOptions*</Pub></mqps>
MQPubLevel	mqps.Pbl	i8	<mqps><Pbl>*publicationLevel*</Pbl></mqps>
MQPubTime	mqpse.Pts	string	<mqps><Pts>*publicationTime*</Pts></mqps>
MQPubSeqNum	mqpse.Seq	i8	<mqps><Seq>*publicationSequenceNumber*</Seq></mqps>
MQPubStrIntData	mqpse.Sid	string	<mqps><Sid>*publicationData*</Sid></mqps>
MQPubFormat	mqpse.Pfmt	i8	<mqps><Pfmt>*messageFormat*</Pfmt></mqps>

Do složky mqps nepřidávejte své vlastní vlastnosti.

Tabulka 19 na stránce 129 zobrazuje úplný seznam názvů vlastností.

Tabulka 19. Složky a vlastnosti MQRFH2 použité produktem JMS				
JMS Název pole	**Java typ**	**Název složky MQRFH2**	**Název vlastnosti**	**Typ/hodnoty**
JMSDestination	Místo určení	jms	Dst	řetězec
JMSExpiration	long	jms	EXP	i8

Tabulka 19. Složky a vlastnosti MQRFH2 použité produktem JMS (pokračování)				
JMS Název pole	**Java typ**	**Název složky MQRFH2**	**Název vlastnosti**	**Typ/hodnoty**
JMSPriority.	celé číslo	jms	PRI	i4
JMSDeliveryMode	celé číslo	jms	Dlv	i4
JMSCorrelationID	Řetězec	jms	CID	řetězec
JMSReplyTo	Místo určení	jms	Rto	řetězec
JMSTimestamp	long	jms	Tms	i8
JMSType.	Řetězec	McC	Typ, Nastavit, Fmt	řetězec
JMSXGroupID	Řetězec	jms	GID	řetězec
JMSXGroupSeq	celé číslo	jms	Pořadí	i4
xxx (definovaný uživatelem)	Libovolný	usr	xxx	jakékoli
		McC	MSD	jms_none jms_text jms_bajtů jms_map jms_stream jms_object

NameValueDélka (MQLONG)
Délka v bajtech datového řetězce NameValue, který bezprostředně následuje za tímto polem délky (nezahrnuje jeho vlastní délku).

Data NameValueData (MQCHARn)
Řetězec s jedním znakem, jehož délka v bajtech je dána předchozím polem NameValueLength. Obsahuje složku obsahující posloupnost vlastností. Každá vlastnost je triplet název/typ/hodnota, který je obsažen v prvku XML, jehož název je název složky, a to takto:

```
<foldername>
triplet1 triplet2 .....  tripletn </foldername>
```

Za uzavírací značkou `</foldername>` může následovat mezery jako doplnění znaků. Každý triplet je zakódován pomocí syntaxe podobné XML:

```
<name dt='datatype'>value</name>
```

Prvek `dt='datatype'` je volitelný a je vynechán pro mnoho vlastností, protože datový typ je předdefinovaný. Je-li zahrnut, musí být před značku `dt=` zahrnut jeden nebo více mezer.

`name`
je název vlastnosti; viz Tabulka 19 na stránce 129.

`datatype`
musí odpovídat, po skládání, jeden z datových typů uvedených v Tabulka 20 na stránce 131.

`value`
je řetězcová reprezentace hodnoty, která má být předána, s použitím definic v produktu Tabulka 20 na stránce 131.

Hodnota null je zakódována pomocí následující syntaxe:

```
<name dt='datatype' xsi:nil='true'></name>
```

Nepoužívejte `xsi:nil='false'`.

Tabulka 20. Datové typy vlastností

Datový typ	**Definice**
řetězec	Libovolné pořadí znaků kromě < a &
typ boolean	Znak `0` nebo 1 (`0` = false, 1 = true)
bin.hex	hexadecimální číslice představující oktety
i1	Číslo, vyjádřené pomocí číslic `0..9`, s volitelným znaménkem (bez zlomků nebo exponentu). Musí ležet v rozsahu -128 až 127 včetně
i2	Číslo, vyjádřené pomocí číslic `0..9`, s volitelným znaménkem (bez zlomků nebo exponentu). Musí ležet v rozsahu -32768 až 32767 včetně
i4	Číslo, vyjádřené pomocí číslic `0..9`, s volitelným znaménkem (bez zlomků nebo exponentu). Musí ležet v rozsahu -2147483648 až 2147483647 včetně
i8	Číslo, vyjádřené pomocí číslic `0..9`, s volitelným znaménkem (bez zlomků nebo exponentu). Musí ležet v rozsahu -9223372036854775808 až 922337
celé číslo	Číslo, vyjádřené pomocí číslic `0..9`, s volitelným znaménkem (bez zlomků nebo exponentu). Musí ležet ve stejném rozsahu jako i8. Tuto hodnotu lze použít místo jednoho z typů systému i *, pokud odesilatel nechce přidružit konkrétní přesnost k vlastnosti
r4	Číslo s pohyblivou řádovou čárkou, velikost < = 3.40282347E+38, > = 1.175E-37 vyjádřený pomocí číslic `0..9`, volitelného znaku, volitelných zlomkových číslic, nepovinného exponentu
r8	Číslo s pohyblivou řádovou čárkou, velikost < = 1.7976931348623E+308, > = 2.225E-307 vyjádřená pomocí číslic `0..9`, volitelného znaku, nepovinného zlomkového čísla, volitelného exponentu

Hodnota řetězce může obsahovat mezery. Musíte použít následující řídicí posloupnosti znaků v řetězcové hodnotě:

- `&` pro znak &
- `<` pro znak <

Můžete použít následující řídicí posloupnosti, které však nejsou povinné:

- `>` pro znak >
- `&apos;` pro znak '
- `&quot;` pro znak "

Pole a vlastnosti produktu JMS s příslušnými poli MQMD
Tyto tabulky zobrazují pole MQMD odpovídající polím záhlaví produktu JMS , vlastnostem produktu JMS a vlastnostem specifickým pro poskytovatele JMS .

Tabulka 21 na stránce 132 vypíše pole záhlaví JMS a Tabulka 22 na stránce 132 vypíše vlastnosti JMS , které jsou mapovány přímo na pole MQMD. Tabulka 23 na stránce 132 obsahuje seznam vlastností specifických pro poskytovatele a polí MQMD, na které jsou mapovány.

Tabulka 21. Mapování polí záhlaví produktu JMS na pole MQMD			
JMS Pole záhlaví	**Java typ**	**Pole MQMD**	**Typ C**
JMSDeliveryMode	celé číslo	Trvání	MQLONG
JMSExpiration	long	Vypršení	MQLONG
JMSPriority.	celé číslo	Priorita	MQLONG
JMSMessageID	Řetězec	MsgID	MQBYTE24
JMSTimestamp	long	PutDate PutTime	MQCHAR8 MQCHAR8
JMSCorrelationID	Řetězec	CorrelId	MQBYTE24

Tabulka 22. Mapování vlastností produktu JMS na pole MQMD			
JMS vlastnost	**Java typ**	**Pole MQMD**	**Typ C**
JMSXUserID	Řetězec	UserIdentifier	MQCHAR12
JMSXAppID	Řetězec	PutApplName	MQCHAR28
JMSXDeliveryCount	celé číslo	BackoutCount	MQLONG
JMSXGroupID	Řetězec	GroupId	MQBYTE24
JMSXGroupSeq	celé číslo	MsgSeqNumber	MQLONG

Tabulka 23. Mapování vlastností specifických pro poskytovatele produktu JMS na pole MQMD			
Vlastnost specifická pro poskytovatele JMS	**Java typ**	**Pole MQMD**	**Typ C**
Výjimka JMS_IBM_Report_Exception	celé číslo	Sestava	MQLONG
JMS_IBM_Report_Expiration.	celé číslo	Sestava	MQLONG
JMS_IBM_Report_COA	celé číslo	Sestava	MQLONG
JMS_IBM_Report_COD.	celé číslo	Sestava	MQLONG
JMS_IBM_Report_PAN.	celé číslo	Sestava	MQLONG
JMS_IBM_Report_NAN.	celé číslo	Sestava	MQLONG
JMS_ID_zprávy_IBM_Report_ID	celé číslo	Sestava	MQLONG

Tabulka 23. Mapování vlastností specifických pro poskytovatele produktu JMS na pole MQMD (pokračování)			
Vlastnost specifická pro poskytovatele JMS	**Java typ**	**Pole MQMD**	**Typ C**
JMS_IBM_Report_Pass_Correl_ID.	celé číslo	Sestava	MQLONG
JMS_IBM_Report_Discard_Msg.	celé číslo	Sestava	MQLONG
JMS_IBM_MsgType .	celé číslo	MsgType	MQLONG
JMS_IBM_Feedback.	celé číslo	Zpětná vazba	MQLONG
JMS_IBM_Format.	Řetězec	Formát"1" na stránce 133	MQCHAR8
Typ JMS_IBM_PutAppl	celé číslo	PutApplType	MQLONG
JMS_IBM_Encoding	celé číslo	Kódování	MQLONG
JMS_IBM_Character_Set.	Řetězec	CodedCharacterSetId "2" na stránce 133	MQLONG
JMS_IBM_PutDate .	Řetězec	PutDate	MQCHAR8
JMS_IBM_PutTime .	Řetězec	PutTime	MQCHAR8
JMS_IBM_Last_Msg_In_Group.	typ boolean	MsgFlags	MQLONG

Poznámka:

1. Formát JMS_IBM_Format představuje formát těla zprávy. To může být definováno aplikací, která nastavuje vlastnost JMS_IBM_Format zprávy (všimněte si, že má 8 znaků omezení) nebo může být standardně formát IBM MQ těla zprávy vhodné pro typ zprávy JMS . JMS_IBM_Format se mapuje na pole Formát MQMD pouze v případě, že zpráva neobsahuje žádné sekce RFH nebo RFH2 . V typické zprávě je mapován na pole Formát záhlaví RFH2 bezprostředně před tělem zprávy.
2. Hodnota vlastnosti JMS_IBM_Character_Set je řetězcová hodnota, která obsahuje znak znakové sady Java ekvivalentní pro číselnou hodnotu CodedCharacterSetId . Pole MQMD CodedCharacterSetId je číselná hodnota, která obsahuje ekvivalent řetězce znakové sady Java specifikovaného vlastností JMS_IBM_Character_Set.

Mapování polí JMS na pole IBM MQ (odchozí zprávy)
Tyto tabulky ukazují, jak jsou pole záhlaví a vlastností JMS mapována do polí MQMD a MQRFH2 v čase send () nebo publish ().

Tabulka 24 na stránce 134 ukazuje, jak jsou pole záhlaví JMS mapována do polí MQMD/RFH2 v čase send () nebo publish (). Tabulka 25 na stránce 134 ukazuje, jak jsou vlastnosti produktu JMS mapovány do polí MQMD/RFH2 v čase odeslání () nebo publikování (). Tabulka 26 na stránce 134 ukazuje, jak jsou vlastnosti specifické pro poskytovatele JMS mapovány na pole MQMD v čase send () nebo publish (),

Pro pole označená objektem zpráv je přenášená hodnota hodnotou, která je zadržena ve zprávě JMS bezprostředně před operací send () nebo publish (). Hodnota ve zprávě JMS je ponechána nezměněná operací.

Pro pole označená Send Method se přiřadí hodnota při provádění metody send () nebo publish () (jakákoli hodnota uchovávané ve zprávě JMS se ignoruje). Hodnota ve zprávě JMS se aktualizuje a zobrazí se použitá hodnota.

Pole označená jako Přijmout nejsou přenesena a jsou ve zprávě ponechána beze změny pomocí funkce send () nebo publish ().

Tabulka 24. Mapování pole odchozí zprávy

Název pole záhlaví JMS	**Pole MQMD použité pro přenos**	**Header**	**Nastavil:**
JMSDestination		MQRFH2	Odeslat metodu
JMSDeliveryMode	Trvání	MQRFH2	Odeslat metodu
JMSExpiration	Vypršení	MQRFH2	Odeslat metodu
JMSPriority.	Priorita	MQRFH2	Odeslat metodu
JMSMessageID	MsgID		Odeslat metodu
JMSTimestamp	PutDate/PutTime		Odeslat metodu
JMSCorrelationID	CorrelId	MQRFH2	Objekt zprávy
JMSReplyTo	QMgr ReplyToQ/ReplyToQMgr	MQRFH2	Objekt zprávy
JMSType.		MQRFH2	Objekt zprávy
JMSRedelivered			Pouze příjem

Poznámka:

1. Pole MQMD CodedCharacterSetId je číselná hodnota, která obsahuje ekvivalent řetězce znakové sady Java specifikovaného vlastností JMS_IBM_Character_Set.

Tabulka 25. Mapování vlastností odchozí zprávy JMS

JMS Název vlastnosti	**Pole MQMD použité pro přenos**	**Header**	**Nastavil:**
JMSXUserID	UserIdentifier		Odeslat metodu
JMSXAppID	PutApplName		Odeslat metodu
JMSXDeliveryCount			Pouze příjem
JMSXGroupID	GroupId	MQRFH2	Objekt zprávy
JMSXGroupSeq	MsgSeqNumber	MQRFH2	Objekt zprávy

Tabulka 26. Odchozí zpráva JMS -mapování vlastností specifické pro poskytovatele

Název vlastnosti specifický pro poskytovatele JMS	**Pole MQMD použité pro přenos**	**Header**	**Nastavil:**
Výjimka JMS_IBM_Report_Exception	Sestava		Objekt zprávy
JMS_IBM_Report_Expiration.	Sestava		Objekt zprávy
JMS_IBM_Report_COA/COD.	Sestava		Objekt zprávy
JMS_IBM_Report_NAN/PAN.	Sestava		Objekt zprávy
JMS_ID_zprávy_IBM_Report_ID	Sestava		Objekt zprávy
JMS_IBM_Report_Pass_Correl_ID.	Sestava		Objekt zprávy

Tabulka 26. Odchozí zpráva JMS -mapování vlastností specifické pro poskytovatele (pokračování)			
Název vlastnosti specifický pro poskytovatele JMS	**Pole MQMD použité pro přenos**	**Header**	**Nastavil:**
JMS_IBM_Report_Discard_Msg.	Sestava		Objekt zprávy
JMS_IBM_MsgType .	MsgType		Objekt zprávy
JMS_IBM_Feedback.	Zpětná vazba		Objekt zprávy
JMS_IBM_Format.	Formát		Objekt zprávy
Typ JMS_IBM_PutAppl	PutApplType		Odeslat metodu
JMS_IBM_Encoding	Kódování		Objekt zprávy
JMS_IBM_Character_Set.	CodedCharacterSetId		Objekt zprávy
JMS_IBM_PutDate .	PutDate		Odeslat metodu
JMS_IBM_PutTime .	PutTime		Odeslat metodu
JMS_IBM_Last_Msg_In_Group.	MsgFlags		Objekt zprávy

Mapování polí záhlaví JMS v souboru send () nebo publish ()
Tyto poznámky se vztahují k mapování polí produktu JMS na funkce send () nebo publish ().

JMSDestination na MQRFH2
Ten je uložen jako řetězec, který serializuje hlavní charakteristiky cílového objektu, takže přijímající JMS může znovu vytvořit ekvivalentní cílový objekt. Pole MQRFH2 je zakódováno jako identifikátor URI (podrobnosti o notaci identifikátoru URI naleznete v části “Uniform resource identifiers (URI)” na stránce 193).

JMSReplyTo na MQMD.ReplyToQ, ReplyToQMgr, MQRFH2 .
Název fronty je zkopírován do MQMD.ReplyToQ a název správce front je zkopírován do polí správce front ReplyToQMgr. Informace o rozšíření místa určení (další užitečné podrobnosti, které jsou uchovány v cílovém objektu) se zkopírují do pole MQRFH2 . Pole MQRFH2 je zakódováno jako identifikátor URI (viz “Uniform resource identifiers (URI)” na stránce 193 . Podrobnosti o notaci identifikátoru URI).

JMSDeliveryMode na MQMD.Persistence
Hodnota JMSDeliveryMode je nastavena metodou send () nebo publish () nebo MessageProducer, pokud objekt Destination Object nepřepíše tento objekt. Hodnota JMSDeliveryMode je mapována na MQMD.Persistence :

- Hodnota JMS PERSISTENT je ekvivalentní hodnotě MQPER_PERSISTENT.
- Hodnota JMS NON_PERSISTENT odpovídá MQPER_NOT_PERSISTENT.

Není-li vlastnost perzistence MQQueue nastavena na hodnotu WMQConstants.WMQ_PER_QDEF, je hodnota režimu doručení kódována také v rámci struktury MQRFH2.

JMSExpirace do/z MQMD.Expiry, MQRFH2 .
JMSExpiration ukládá čas na vypršení platnosti (součet aktuálního času a doby životnosti), zatímco MQMD ukládá čas k životu. Také hodnota JMSExpiration je v milisekundách, ale MQMD.Expiry je v desetinách sekundy.

- Pokud metoda send () nastavuje neomezenou dobu k životu, MQMD.Expiry je nastaveno na hodnotu MQEI_UNLIMITED a v rámci struktury MQRFH2není zakódována žádná hodnota JMSExpiration.
- Pokud metoda send () nastaví čas životnosti, který je menší než 214748364.7 sekund (asi 7 let), je doba života uložena v MQMD.Expirya doba vypršení platnosti (v milisekundách) je zakódována jako hodnota i8 v MQRFH2.
- Pokud metoda send () nastaví dobu života větší než 214748364.7 sekund, MQMD.Expiry je nastaveno na hodnotu MQEI_UNLIMITED. Doba vypršení platnosti true v milisekundách je zakódována jako hodnota i8 v MQRFH2.

JMSPriority až MQMD.Priority
Přímo namapujte hodnotu JMSPriority (0-9) na hodnotu priority MQMD (0-9). Je-li položka JMSPriority nastavena na jinou než výchozí hodnotu, bude úroveň priority také zakódována v MQRFH2.

JMSMessageID z MQMD.MessageID
Všechny zprávy odeslané z produktu JMS mají jedinečné identifikátory zpráv přiřazené produktem IBM MQ. Přiřazená hodnota je vrácena v deskriptoru MQMD.MessageId po volání MQPUT a předává se zpět aplikaci v poli JMSMessageID . IBM MQ messageId je 24bajtová binární hodnota, zatímco hodnota JMSMessageID je řetězec. Objekt JMSMessageID se skládá z binární hodnoty messageId převedené na posloupnost 48 hexadecimálních znaků s předponou ID znaků:. Produkt JMS poskytuje nápovědu, která může být nastavena tak, aby zakázala vytváření identifikátorů zpráv. Tento pokyn je ignorován a ve všech případech je přiřazen jedinečný identifikátor. Jakákoli hodnota, která je nastavena do pole JMSMessageID před přepsáním funkce send ().

Vyžadujete-li schopnost určení MQMD.MessageID, můžete to provést s jedním z přípon IBM MQ JMS popsaných v "Čtení a zápis deskriptoru zpráv z aplikace IBM MQ classes for JMS" na stránce 210.

JMSTimstamp na MQRFH2 .
Během odeslání je pole JMSTimstamp nastaveno v souladu s hodinami prostředí JVM. Tato hodnota je nastavena na MQRFH2. Jakákoli hodnota, která je nastavena do pole JMSTimstamp před přepsáním funkce send (), je přepsána. Viz také vlastnosti JMS_IBM_PutDate a JMS_IBM_PutTime .

JMSType na MQRFH2 .
Tento řetězec je nastaven do pole MQRFH2 mcd.Type . Pokud se nachází ve formátu identifikátoru URI, může také ovlivnit pole mcd.Set a mcd.Fmt .

JMSCorrelationID na MQMD.CorrelId, MQRFH2 .
JMSCorrelationID může mít jednu z následujících možností:

ID zprávy specifické pro poskytovatele
Jedná se o identifikátor zprávy z dříve odeslané nebo přijaté zprávy, a proto by měl být řetězec 48 hexadecimálních hexadecimálních číslic, který má předponu `ID:` Předpona je odebrán, zbývající znaky jsou převedeny na binární, a pak jsou nastaveny do deskriptoru MQMD.CorrelId . Hodnota CorrelId není zakódována v MQRFH2.

Poskytovatel-nativní bajtová hodnota []
Hodnota je zkopírována do deskriptoru MQMD.CorrelId -vyplněno hodnotami null nebo zkráceno na 24 bajtů, je-li to nutné. Hodnota CorrelId není zakódována v MQRFH2.

Řetězec specifický pro aplikaci
Hodnota je zkopírována do MQRFH2. Prvních 24 bajtů řetězce, ve formátu UTF8 , jsou zapsány do deskriptoru MQMD.CorrelID.

Pole vlastností mapování JMS
Tyto poznámky odkazují na mapování polí vlastností produktu JMS ve zprávách produktu IBM MQ .

JMSXUserID z MQMD UserIdentifier
Hodnota JMSXUserID je nastavena na návrat z volání odeslání.

JMSXAppID z názvu MQMD PutAppl
JSMXAppID je nastaven na návrat z volání odeslání.

JMSXGroupID na MQRFH2 (dvoubodový).
Pro zprávy typu point-to-point se zkopíruje JMSXGroupID do pole MQMD GroupID . Pokud se JMSXGroupID spustí s ID prefixu:, je převeden na binární. Jinak je zakódován jako řetězec UTF8 . Hodnota je polstrovaná nebo osekána, je-li to nutné, na délku 24 bajtů. Je nastaven příznak MQMF_MSG_IN_GROUP.

JMSXGroupID na MQRFH2 (publish/subscribe).
U zpráv typu publikování/odběr se JMSXGroupID zkopíruje do struktury MQRFH2 jako řetězec.

JMSXGroupSeq MQMD MsgSeqČíslo (point-to-point)
Pro zprávy typu point-to-point je proměnná JMSXGroupSeq zkopírována do pole Číslo MQMD MsgSeq. Je nastaven příznak MQMF_MSG_IN_GROUP.

JMSXGroupSeq MQMD MsgSeqČíslo (publish/subscribe)
U zpráv typu publikování/odběr se JMSXGroupSeq zkopíruje do MQRFH2 jako i4.

Mapování polí specifických pro poskytovatele JMS
Následující poznámky odkazují na mapování polí specifických pro poskytovatele JMS na zprávy produktu IBM MQ .

JMS_IBM_Report_*XXX* na sestavu MQMD
Aplikace JMS může nastavit volby sestavy MQMD s použitím následujících vlastností JMS_IBM_Report_*XXX* . Jediný deskriptor MQMD je mapován na několik vlastností JMS_IBM_Report_*XXX* . Aplikace musí nastavit hodnotu těchto vlastností na standardní konstanty produktu IBM MQ MQRO_ (zahrnuté v com.ibm.mq.MQC). Například, chcete-li požádat o CHSK s úplnými daty, aplikace musí nastavit parametr JMS_IBM_Report_COD na hodnotu CMQC.MQRO_COD_WITH_FULL_DATA.

Výjimka JMS_IBM_Report_Exception

MQRO_EXCEPTION nebo
MQRO_EXCEPTION_WITH_DATA nebo
MQRO_EXCEPTION_WITH_FULL_DATA

JMS_IBM_Report_Expiration.

MQRO_EXPIRATION nebo
MQRO_EXPIRATION_WITH_DATA nebo
MQRO_EXPIRATION_WITH_FULL_DATA

JMS_IBM_Report_COA

MQRO_COA nebo
MQRO_COA_WITH_DATA nebo
MQRO_COA_WITH_FULL_DATA

JMS_IBM_Report_COD.

MQRO_COD nebo
MQRO_COD_WITH_DATA nebo
MQRO_COD_WITH_FULL_DATA

JMS_IBM_Report_PAN.
MQRO_PAN

JMS_IBM_Report_NAN.
MQRO_NAN

JMS_ID_zprávy_IBM_Report_ID
MQRO_PASS_MSG_ID

JMS_IBM_Report_Pass_Correl_ID.
ID_KOLEKCE_MQRO_PASS_RELACE_

JMS_IBM_Report_Discard_Msg.
MQRO_DISCARD_MSG

JMS_IBM_MsgType to MQMD MsgType .
Mapy hodnot přímo na MQMD MsgType. Pokud aplikace nenastavila explicitní hodnotu parametru JMS_IBM_MsgType, použije se výchozí hodnota. Tato výchozí hodnota je určena následujícím způsobem:

- Je-li vlastnost JMSReplyTo nastavena na cíl fronty IBM MQ , hodnota MSGType je nastavena na hodnotu MQMT_REQUEST
- Pokud vlastnost JMSReplyTo není nastavena nebo je nastavena na jinou hodnotu než cíl fronty produktu IBM MQ , hodnota MsgType je nastavena na hodnotu MQMT_DATAGRAM.

JMS_IBM_Feedback k zpětné vazbě MQMD
Mapy hodnot přímo na zpětnou vazbu MQMD.

JMS_IBM_Format na formát MQMD.
Mapy hodnot přímo do formátu MQMD.

JMS_IBM_Encoding na kódování MQMD.
Je-li tato vlastnost nastavena, přepíše číselné kódování cílové fronty nebo tématu.

JMS_IBM_Character_Set to MQMD CodedCharacterSetId .
Je-li nastavena, potlačí tato vlastnost vlastnost kódované znakové sady cílové fronty nebo tématu.

JMS_IBM_PutDate z MQMD PutDate
Hodnota této vlastnosti je nastavena během odesílání přímo z pole PutDate v MQMD. Jakákoli hodnota, která je nastavena do vlastnosti JMS_IBM_PutDate před přepsáním odeslání. Toto pole je řetězec osmi znaků, ve formátu data IBM MQ RRRRMMDD. Tuto vlastnost lze použít s vlastností JMS_IBM_PutTime k určení času, kdy byla zpráva vložena do správce front.

JMS_IBM_PutTime z MQMD PutTime .
Hodnota této vlastnosti je nastavena během odeslání přímo z pole PutTime v MQMD. Jakákoli hodnota, která je nastavena do vlastnosti JMS_IBM_PutTime před přepsáním odeslání. Toto pole je řetězec osmi znaků, ve formátu IBM MQ Čas HHMMSSTH. Tuto vlastnost lze použít spolu s vlastností JMS_IBM_PutDate k určení času, kdy byla zpráva vložena do správce front.

JMS_IBM_Last_Msg_In_Group-MQMD MsgFlags .
Pro systém zpráv typu point-to-point se tato logická hodnota mapuje na příznak MQMF_LAST_MSG_IN_GROUP v poli MQMD MsgFlags . Obvykle se používá s vlastnostmi JMSXGroupID a JMSXGroupSeq pro označení starší aplikace produktu IBM MQ , že tato zpráva je poslední ve skupině. Tato vlastnost je ignorována při publikování/odběru zpráv.

Mapování polí IBM MQ na pole JMS (příchozí zprávy)
Tyto tabulky zobrazují, jak jsou pole záhlaví a vlastnosti JMS mapována do polí MQMD a MQRFH2 v čase get () nebo receive ().

Příkaz Tabulka 27 na stránce 138 zobrazuje, jak jsou pole záhlaví JMS mapována na pole MQMD/MQRFH2 v čase get () nebo receive (). Příkaz Tabulka 28 na stránce 139 zobrazuje, jak jsou pole vlastností JMS mapována na pole MQMD/MQRFH2 v čase get () nebo receive (). Příkaz Tabulka 29 na stránce 139 ukazuje, jak jsou mapovány vlastnosti specifické pro poskytovatele JMS .

Tabulka 27. Mapování pole záhlaví příchozí zprávy JMS

Název pole záhlaví JMS	**Pole MQMD je načteno z**	**Pole MQRFH2 načteno z**
JMSDestination		jms.Dst nebo mqps.Top "1" na stránce 139
JMSDeliveryMode	Trvání"2" na stránce 139	jms.Dlv "2" na stránce 139
JMSExpiration		jms.Exp
JMSPriority.	Priorita	
JMSMessageID	MsgID	
JMSTimestamp	PutDate "2" na stránce 139 PutTime "2" na stránce 139	jms.Tms "2" na stránce 139
JMSCorrelationID	CorrelId "2" na stránce 139	jms.Cid "2" na stránce 139
JMSReplyTo	ReplyToQ "2" na stránce 139 ReplyTosprávce front "2" na stránce 139	jms.Rto "2" na stránce 139
JMSType.		mcd.Type, mcd.Set, mcd.Fmt
JMSRedelivered	BackoutCount	

Poznámka:

1. Je-li nastavena hodnota jms.Dst a mqps.Top , použije se hodnota ve struktuře jms.Dst .
2. Pro vlastnosti, které mohou mít hodnoty načtené ze MQRFH2 nebo MQMD, je-li k dispozici obojí, je použito nastavení v MQRFH2 .
3. Hodnota vlastnosti JMS_IBM_Character_Set je řetězcová hodnota, která obsahuje znak znakové sady Java ekvivalentní pro číselnou hodnotu CodedCharacterSetId .

Tabulka 28. Mapování vlastností příchozích zpráv

JMS Název vlastnosti	**Pole MQMD je načteno z**	**Pole MQRFH2 načteno z**
JMSXUserID	UserIdentifier	
JMSXAppID	PutApplName	
JMSXDeliveryCount	BackoutCount	
JMSXGroupID	GroupId “1” na stránce 139	jms.Gid “1” na stránce 139
JMSXGroupSeq	MsgSeqČíslo “1” na stránce 139	jms.Seq “1” na stránce 139

Poznámka:

1. Pro vlastnosti, které mohou mít hodnoty načtené ze MQRFH2 nebo MQMD, je-li k dispozici obojí, je použito nastavení v MQRFH2 . Vlastnosti jsou nastaveny z hodnot MQMD pouze v případě, že jsou nastaveny příznaky zpráv MQMF_MSG_IN_GROUP nebo MQMF_LAST_MSG_IN_GROUP.

Tabulka 29. Mapování vlastností produktu Incoming Message Provider-specifické JMS

JMS Název vlastnosti	**Pole MQMD je načteno z**	**Pole MQRFH2 načteno z**
Výjimka JMS_IBM_Report_Exception	Sestava	
JMS_IBM_Report_Expiration.	Sestava	
JMS_IBM_Report_COA	Sestava	
JMS_IBM_Report_COD.	Sestava	
JMS_IBM_Report_PAN.	Sestava	
JMS_IBM_Report_NAN.	Sestava	
JMS_IBM_Report_ Pass_Msg_ID.	Sestava	
JMS_IBM_Report_Pass_Correl_ID.	Sestava	
JMS_IBM_Report_Discard_Msg.	Sestava	
JMS_IBM_MsgType .	MsgType	
JMS_IBM_Feedback.	Zpětná vazba	
JMS_IBM_Format.	Formát	
Typ JMS_IBM_PutAppl	PutApplType	
JMS_IBM_Encoding “1” na stránce 140 .	Kódování	
JMS_IBM_Character_Set “1” na stránce 140 .	CodedCharacterSetId	
JMS_IBM_PutDate .	PutDate	
JMS_IBM_PutTime .	PutTime	
JMS_IBM_Last_Msg_In_Group.	MsgFlags	

1. Nastavit pouze v případě, že příchozí zpráva je bajtová zpráva.

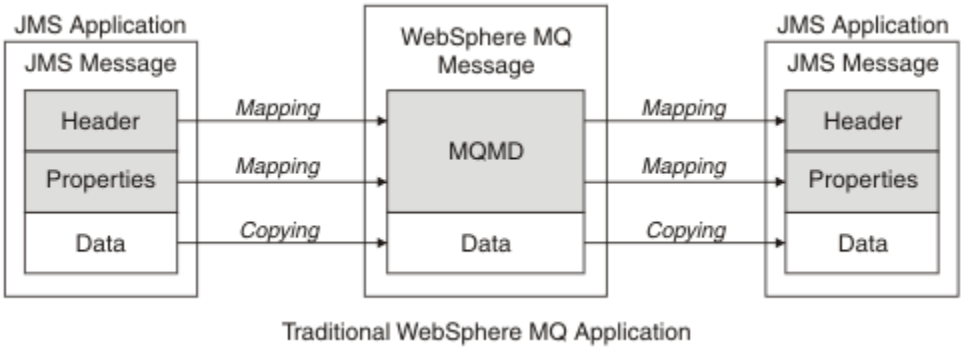
Výměna zpráv mezi aplikací JMS a tradiční aplikací IBM MQ
Toto téma popisuje, co se stane, když aplikace JMS vymění zprávy s tradiční aplikací produktu IBM MQ , která nemůže zpracovat záhlaví MQRFH2 .

Obrázek 11 na stránce 140 zobrazuje mapování.

Administrátor informuje o tom, že aplikace JMS komunikuje s tradiční aplikací IBM MQ nastavením vlastnosti TARGCLIENT cíle na hodnotu *MQ*. To označuje, že se nemá vytvořit žádná hlavička MQRFH2 . Není-li tato akce provedena, přijímající aplikace musí být schopna zpracovat záhlaví MQRFH2 .

Mapování z produktu JMS na MQMD zaměřené na tradiční aplikaci produktu IBM MQ je stejné jako mapování z produktu JMS na MQMD zacílené na aplikaci produktu JMS . Pokud příkaz IBM MQ classes for JMS přijme zprávu IBM MQ s polem MQMD `Format` nastaveným na hodnotu jinou než MQFMT_RFH2, budou data přijata z aplikace, která není typuJMS . Je-li formát MQFMT_STRING, bude zpráva přijata jako textová zpráva JMS . Jinak se přijme jako zpráva JMS bajtů. Protože zde není žádný MQRFH2, lze obnovit pouze vlastnosti JMS , které jsou přeneseny v MQMD.

Pokud produkt IBM MQ classes for JMS obdrží zprávu, která nemá záhlaví MQRFH2 , je vlastnost TARGCLIENT objektu Queue nebo Topic odvozeného z pole záhlaví JMSReplyTo ve zprávě standardně nastavena na hodnotu MQ . To znamená, že zpráva odpovědi odeslaná do fronty nebo tématu také nemá záhlaví MQRFH2 . Toto chování včetně záhlaví MQRFH2 ve zprávě odpovědi můžete vypnout pouze v případě, že původní zpráva má záhlaví MQRFH2 , a to nastavením vlastnosti TARGCLIENTMATCHING továrny na připojení na NO.



Obrázek 11. Jak se zprávy produktu JMS transformují na zprávy produktu IBM MQ bez záhlaví MQRFH2 ,

Tělo zprávy produktu JMS
Toto téma obsahuje informace o kódování samotného těla zprávy. Kódování závisí na typu zprávy JMS .

ObjectMessage
ObjectMessage je objekt serializovaný běhovým prostředím Java běžným způsobem.

TextMessage
Řetězec TextMessage je zakódovaný řetězec. U odchozí zprávy je řetězec zakódován ve znakové sadě, která je dána cílovým objektem. Výchozí hodnota je kódování UTF8 (kódování UTF8 začíná prvním znakem zprávy; na začátku není žádné pole délky). Je však možné zadat jakoukoli jinou znakovou sadu podporovanou produktem IBM MQ classes for JMS. Tyto znakové sady se používají hlavně tehdy, když posíláte zprávu do aplikace, která není typuJMS .

Je-li znaková sada dvoubajtová sada (včetně UTF16), určuje se pořadí bajtů ve specifikaci kódování celých čísel cílového objektu.

Příchozí zpráva je interpretována pomocí znakové sady a kódování, které jsou uvedeny ve zprávě samotné. Tyto specifikace jsou v posledním záhlaví IBM MQ (nebo MQMD, pokud nejsou žádná záhlaví). V případě zpráv produktu JMS je posledním záhlavím obvykle hodnota MQRFH2.

BytesMessage
BytesMessage je standardně posloupnost bajtů, jak je definováno specifikací JMS 1.0.2 a přidruženou dokumentací Java .

Pro odchozí zprávu, která byla sestavena aplikací samotnou, lze vlastnost kódování cílového objektu použít k potlačení kódování celých čísel a polí s pohyblivou řádovou čárkou obsažených ve zprávě. Můžete například požadovat, aby byly hodnoty s pohyblivou řádovou čárkou uloženy v systému S/390 namísto formátu IEEE).

Příchozí zpráva je interpretována s použitím numerického kódování určeného ve zprávě samotné. Tato specifikace je v posledním záhlaví produktu IBM MQ (nebo MQMD, pokud neexistují záhlaví). V případě zpráv produktu JMS je posledním záhlavím obvykle hodnota MQRFH2.

Pokud je přijata hodnota BytesMessage a je znovu odeslána bez úpravy, její tělo je přenášen bajt po bajtech, jak bylo přijato. Vlastnost kódování cílového objektu nemá žádný vliv na tělo. Jediná entita podobná řetězci, která může být odeslána explicitně v BytesMessage , je řetězec UTF8 . To je zakódováno ve formátu Java UTF8 a začíná se 2bajtovým polem délky. Vlastnost znakové sady cílového objektu nemá žádný vliv na kódování odchozích BytesMessage. Hodnota znakové sady v příchozí zprávě IBM MQ nemá žádný vliv na interpretaci této zprávy ve tvaru JMS BytesMessage.

Aplikace Nona-Java pravděpodobně nerozeznají kódování Java UTF8 . Proto, aby aplikace JMS odeslala zprávu BytesMessage , která obsahuje textová data, musí sama aplikace převést své řetězce na pole bajtů a zapsat tato bajtová pole do BytesMessage.

MapMessage

MapMessage je řetězec obsahující triplety XML název/typ/hodnoty kódované jako:

```
<map>
  <elt name="elementname1" dt="datatype1">value1</elt>
  <elt name="elementname2" dt="datatype2">value2</elt>
  ...
</map>
```

kde `datatype` je jeden z datových typů uvedených v Tabulka 20 na stránce 131. Výchozí datový typ je `string`, a proto je atribut `dt="string"` vynechán pro řetězcové prvky.

Znaková sada použitá ke kódování nebo interpretaci řetězce XML, který tvoří tělo mapy zpráv, je určen podle pravidel platných pro textovou zprávu.

Verze produktu IBM MQ classes for JMS starší než 5.3 zakódované tělo mapy zpráv v následujícím formátu:

```
<map>
  <elementname1 dt="datatype1">value1</elementname1>
  <elementname2 dt="datatype2">value2</elementname2>
  ...
</map>
```

IBM MQ classes for JMS 5.3 a pozdější mohou interpretovat oba formáty, ale verze produktu IBM MQ classes for JMS starší než 5.3 nemohou interpretovat aktuální formát.

Pokud aplikace potřebuje odeslat mapové zprávy do jiné aplikace, která používá verzi produktu IBM MQ classes for JMS starší než 5.3, musí odesílající aplikace volat metodu továrny připojení `setMapNameStyle(WMQConstants.WMQ_MAP_NAME_STYLE_COMPATIBLE)` , aby určoval, že zprávy mapování jsou odesílány v předchozím formátu. Při výchozím nastavení jsou všechny zprávy mapy odesílány v aktuálním formátu.

StreamMessage

StreamMessage je jako zpráva mapy, ale bez názvů prvků:

```
<stream>
  <elt dt="datatype1">value1</elt>
  <elt dt="datatype2">value2</elt>
  ...
</stream>
```

kde `datatype` je jeden z datových typů uvedených v Tabulka 20 na stránce 131. Výchozí datový typ je `string`, a proto je atribut `dt="string"` vynechán pro řetězcové prvky.

Znaková sada použitá ke kódování nebo interpretaci řetězce XML, který vytváří tělo StreamMessage , je určen podle pravidel, která se použijí na TextMessage.

Pole MQRFH2.format je nastaveno takto:

MQFMT_NONE
pro ObjectMessage, BytesMessagenebo zprávy bez těla.

ŘETĚZEC MQFMT_STRING
pro TextMessage, StreamMessage, nebo MapMessage.

Převod zpráv produktu JMS
Převod dat zpráv v produktu JMS se provádí při odesílání a přijímání zpráv. Příkaz IBM MQ provádí většinu automatického převodu dat automaticky. Převádí text a číselná data při přenosu zprávy mezi aplikacemi produktu JMS . Text se převede při výměně `JMSTextMessage` mezi aplikací JMS a aplikací IBM MQ .

Plánujete-li provádět složitější výměny zpráv, následující témata vás zajímají. Komplexní výměny zpráv zahrnují:

- Přenos netextových zpráv mezi aplikací IBM MQ a aplikací JMS .
- Výměna textových dat v bajtovém formátu.
- Probíhá převod textu ve vaší aplikaci.

JMS Data zprávy

Převod dat je nutný pro výměnu textových a číselných dat mezi aplikacemi, a to i mezi dvěma aplikacemi produktu JMS . Interní reprezentace textu a čísel musí být zakódována, aby bylo možné je přenést ve zprávě. Kódování vynutí rozhodnutí o tom, jak jsou čísla a text představeny. Produkt IBM MQ spravuje kódování textu a čísel ve zprávách obslužného programu JMS , s výjimkou produktu `JMSObjectMessage`, viz “JMSObjectMessage” na stránce 149. Používá tři atributy zpráv. Uvedené tři atributy jsou `CodedCharacterSetId`, `Encoding`a `Format`.

Tyto tři atributy zpráv jsou obvykle uloženy v záhlaví JMS , `MQRFH2`, v polích zprávy JMS . Je-li typ zprávy `MQ`, nikoli `JMS` typu zprávy, jsou atributy uloženy v deskriptoru zpráv, `MQMD`. Atributy se používají pro převod dat zprávy produktu JMS . Data zprávy produktu JMS se přenášejí v části dat zprávy ve zprávě IBM MQ .

JMS Vlastnosti zprávy

Vlastnosti zprávy JMS , jako například `JMS_IBM_CHARACTER_SET`, jsou vyměňovány v části záhlaví `MQRFH2` zprávy JMS , pokud nebyla zpráva odeslána bez `MQRFH2`. Bez `MQRFH2`lze odeslat pouze `JMSTextMessage` a `JMSBytesMessage` . Je-li vlastnost JMS uložena jako vlastnost zprávy IBM MQ v deskriptoru zprávy, `MQMD`, je převedena jako součást převodu `MQMD` . Je-li vlastnost JMS uložena v produktu `MQRFH2`, je uložena ve znakové sadě zadané argumentem `MQRFH2.NameValueCCSID`. Je-li zpráva odeslána nebo přijata, jsou vlastnosti zprávy převedeny na jejich interní reprezentaci v prostředí JVM a z jejich interní reprezentace. Převod je nastaven na a ze znakové sady deskriptoru zpráv nebo `MQRFH2.NameValueCCSID`. Číselná data jsou převedena na text.

Převod zpráv produktu JMS

Následující témata obsahují příklady a úlohy, které jsou užitečné v případě, že plánujete výměnu složitějších zpráv, které vyžadují konverzi.

Metody konverze zpráv produktu JMS
Návrháři aplikací produktu JMS jsou otevřeny pro řadu přístupů k převodu dat. Tyto přístupy se nevylučují; některé aplikace budou pravděpodobně používat kombinaci těchto přístupů. Pokud si aplikace vyměňuje pouze text nebo si vyměňuje zprávy pouze s jinými aplikacemi produktu JMS , obvykle nepovažujete převod dat za normální. Převod dat se provádí automaticky pro vás, příkazem IBM MQ.

Můžete se zeptat na několik otázek ohledně toho, jak přistupovat ke konverzi zpráv:

Je třeba přemýšlet o přestavbě zpráv vůbec?
V některých případech, jako je například JMS pro přenos zpráv JMS a pro výměnu textových zpráv s programy IBM MQ , provádí produkt IBM MQ pro vás nezbytné převody automaticky. Možná budete chtít řídit konverzi dat z důvodu výkonu nebo si můžete vyměňovat komplexní zprávy, které mají předdefinovaný formát. V takových případech, jako je tato, musíte rozumět převodu zpráv a přečtěte si následující témata.

Jaký druh konverze je tam?
Existují čtyři hlavní typy převodu, které jsou vysvětleny v následujících sekcích:

1. “Převod dat klienta JMS” na stránce 143
2. “Převod dat aplikace” na stránce 144
3. “Převod dat správce front” na stránce 144
4. “Převod dat kanálu zpráv” na stránce 145

Kde by měl být proveden převod?
Část “Výběr přístupu k převodu zpráv: receiver makes good” na stránce 145popisuje běžný přístup k "přijímači je dobrý". Volba "Receiver makes good" se vztahuje také na převod dat produktu JMS .

Převod dat klienta JMS

JMS klient[1]konverze dat je konverze primitiv Java a objektů do bajtů ve zprávě JMS , jak je odeslána do cíle, a převod zpět znovu, když je přijata. Převod dat klienta JMS používá metody tříd produktu `JMSMessage` . Metody jsou vypsány podle typu třídy `JMSMessage` v Tabulka 30 na stránce 146.

Převod na a z vnitřní reprezentace čísel a textu prostředí JVM se provádí pro metody read, get, set a write. Konverze se provede, když se odešle zpráva, a když je některá z metod read nebo get volána na přijatou zprávu.

Kódová stránka a numerické kódování použité k zápisu nebo nastavení obsahu zprávy jsou definovány jako atributy místa určení. Kódu místa určení a číselné kódování lze změnit administrativně. Aplikace může také přepsat cílovou kódovou stránku a kódování nastavením vlastností zprávy, které řídí zápis nebo nastavení obsahu zprávy.

Chcete-li převést kódování čísel při odeslání zprávy produktu `JMSBytesMessage` na místo určení, které není definováno jako kódování `Native` , je třeba před odesláním zprávy nastavit vlastnost zprávy `JMS_IBM_ENCODING` . Pokud sledujete vzor "receiver makes good" , nebo pokud si vyměňujete zprávy mezi aplikacemi produktu JMS , nemusí být aplikace nastavena na `JMS_IBM_ENCODING`. Ve většině případů můžete opustit vlastnost `Encoding` jako `Native`.

Pro zprávy `JMSStreamMessage`, `JMSMapMessage`a `JMSTextMessage` se používají vlastnosti identifikátoru znakové sady místa určení. Kódování je při odesílání ignorováno, protože čísla jsou zapsána v textovém formátu. Aplikační program klienta JMS nemusí před odesláním zprávy, pokud se má použít vlastnost cílové znakové sady, nastavit `JMS_IBM_CHARACTER_SET` .

Chcete-li získat data ve zprávě, aplikace volá metodu čtení nebo získání zprávy produktu JMS . Metody odkazují na kódovou stránku a kódování definované v předchozím záhlaví zprávy, aby bylo možné správně vytvořit primitiva a objekty produktu Java .

Převod dat klienta JMS vyhovuje potřebám většiny aplikací produktu JMS , které si vyměňují zprávy mezi jedním klientem produktu JMS a druhou. Nekókód explicitního převodu dat nekódujete. Nepoužíváte třídu `java.nio.charset.Charset` , která se obvykle používá při zápisu textu do souboru. Metody `writeString` a `setString` provádí konverzi pro vás.

Další informace o převodu dat klienta JMS naleznete v tématu “Převod a kódování zpráv klienta JMS” na stránce 155.

[1] "KlientJMS" odkazuje na IBM MQ classes for JMS , které implementuje rozhraní JMS , které běží buď v režimu klienta, nebo vazby.

Převod dat aplikace

Klientská aplikace JMS může provádět explicitní konverzi znakových dat pomocí třídy `java.nio.charset.Charset` ; viz příklady v Obrázek 14 na stránce 148 a Obrázek 15 na stránce 148. Řetězcová data se převedou na bajty pomocí metody `getBytes` a odesílají se jako bajty. Bajty jsou převedeny zpět do textu pomocí konstruktoru `String` , který přijímá bajtové pole a `Charset`. Znaková data se konvertují pomocí metod `encode` a `decode` `Charset` . Obvykle je zpráva odeslána nebo přijata jako `JMSBytesMessage`, protože část zprávy `JMSBytesMessage` neobsahuje nic jiného než data zapsaná aplikací.[2]bajtů můžete také odesílat a přijímat pomocí produktů `JMSStreamMessage`, `JMSMapMessage`nebo `JMSObjectMessage`.

Nejsou k dispozici žádné metody Java pro kódování a dekódování bajtů, které obsahují numerická data reprezentovaná v různých formátech kódování. Numerická data jsou kódována a dekódována automaticky pomocí číselných metod čtení a zápisu `JMSMessage` . Metody čtení a zápisu používají hodnotu atributu `JMS_IBM_ENCODING` dat zprávy.

Typickým použitím pro převod dat aplikací je to, že klient JMS odesílá nebo přijímá formátovanou zprávu z aplikace, která není typuJMS . Formátovaná zpráva obsahuje text, číselná data a bajtová data uspořádaná podle délky datových polí. Pokud aplikace mimo aplikaciJMS neurčila formát zprávy jako "MQSTR ", je tato zpráva vytvořena jako `JMSBytesMessage`. Chcete-li přijímat formátovaná data zprávy v produktu `JMSBytesMessage` , musíte volat posloupnost metod. Metody musí být volány ve stejném pořadí, v jakém byla pole zapsána do zprávy. Jsou-li pole numerická, musíte znát kódování a délku číselných dat. Pokud některá z polí obsahují bajtová nebo textová data, musíte znát délku libovolných bajtových dat ve zprávě. Existují dva způsoby, jak převést naformátovanou zprávu do objektu Java , který se snadno používá.

1. Sestavte třídu Java odpovídající záznamu, abyste zapouzdřují čtení a zapisovali zprávu. Přístup k datům v záznamu je s metodami get a set třídy.
2. Zkonstruujte třídu Java odpovídající záznamu rozšířením třídy `com.ibm.mq.headers` . Přístup k datům ve třídě je s přístupoovými mechanizmy specifickými pro daný typ formuláře `getStringValue(`*`fieldname`*`);`

Viz “Výměna formátovaného záznamu s aplikací jinou nežJMS” na stránce 163.

Převod dat správce front

V produktu IBM MQ 7.0může být převod kódové stránky proveden správcem front, když klientský program JMS dostane zprávu. Konverze je stejná jako konverze provedená pro program v jazyce C. Programové sady C nastaví `MQGMO_CONVERT` jako volbu parametru `MQGET` `GetMsgOpts` ; viz Obrázek 13 na stránce 148. Správce front provádí převod pro klientský program JMS , který přijímá zprávu, je-li cílová vlastnost `WMQ_RECEIVE_CONVERSION` nastavena na hodnotu `WMQ_RECEIVE_CONVERSION_QMGR`, může klientský program JMS také nastavit vlastnost cíle; viz Obrázek 12 na stránce 145.

Před verzí 7.0byly konverze vždy provedeny klientem JMS . Převod dat klienta JMS je omezen na převod posloupností čísel a textu typu a délky, které jsou známy klientovi JMS . Nelze konvertovat datové struktury; viz “Výměna formátovaného záznamu s aplikací jinou nežJMS” na stránce 163. Do opravy FixPack 7.0.1.5v produktu 7.0, pokud může být převod proveden správcem front, je tento převod vždy prováděn správcem front. Počínaje verzí 7.0.1.5 se výchozí chování převodu vrací na stejné hodnoty jako 6.0a všechny převody jsou prováděny klientem JMS . Od verze 7.0.1.5nebo 7.0.1.4 s opravou APAR IC72897můžete nastavit novou cílovou volbu, `WMQ_RECEIVE_CONVERSION`, chcete-li řídit, kde je prováděna konverze, a `WMQ_RECEIVE_CCSID`, abyste nastavili cílovou kódovou stránku, viz Obrázek 12 na stránce 145.

[2] Jedna výjimka: Data zapsaná pomocí `writeUTF` začíná na 2 bajtové délce pole

```
((MQDestination)destination).setIntProperty(
              WMQConstants.WMQ_RECEIVE_CONVERSION,
              WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR);
```

Nebo,

```
((MQDestination)destination).setReceiveConversion
              (WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR);
```

Obrázek 12. Povolit převod dat správce front

Hlavní přínos převodu správce front je dodáván při výměně zpráv s aplikacemi jinými nežJMS . Je-li definováno pole `Format` ve zprávě a cílová znaková sada nebo kódování se liší od zprávy, správce front provede převod dat pro cílovou aplikaci, pokud ji aplikace požaduje. Správce front převádí data zprávy zformátovaná podle jednoho z předdefinovaných typů zpráv produktu IBM MQ , jako je záhlaví CICS bridge (`MQCIH`). Je-li pole `Format` definováno uživatelem, vyhledá správce front ukončení převodu dat s názvem uvedeným v poli `Format` .

Převod dat správce front se používá k nejlepšímu účinku se vzorem návrhu "receiver makes good" . Odesílající klient JMS nemusí provádět převod. Přijímací program, který není produktemJMS , závisí na ukončení konverze, aby se zajistilo, že zpráva bude doručena v požadované kódové stránce a kódování. Při odeslání klienta JMS a jiného příjemce nežJMS se tento příklad vztahuje na IBM MQ před a post-V7.0. Při použití produktu IBM MQ 7.0lze ukončovací program pro převod také volat pro přijímající program JMS .

Můžete vytvořit uživatelskou proceduru pro převod dat pomocí obslužného programu ukončení převodu dat **`crtmqcvx`**, chcete-li povolit správci front převést vaše vlastní formátovaná data záznamu. Můžete sestavit váš vlastní formát záznamu, použít `com.ibm.mq.headers` pro přístup k němu jako třídu Java a použít vlastní uživatelskou proceduru konverze k jeho převodu. V systému z/OS se obslužný program nazývá **`CSQUCVX`**a v systému IBM i **`CVTMQMDTA`**. Viz “Výměna formátovaného záznamu s aplikací jinou nežJMS” na stránce 163.

Převod dat kanálu zpráv

Kanály IBM MQ Sender, Server, Cluster-receiver a Cluster-sender mají možnost převodu zpráv, `CONVERT`. Obsah zprávy lze volitelně převést, když se odešle zpráva. Konverze probíhá na odesílajícím konci kanálu. Definice příjemce klastru se používá k automatickému definování odpovídajícího odesílacího kanálu klastru.

Převod dat podle kanálů zpráv se obvykle používá v případě, že není možné použít jiné formy převodu.

Výběr přístupu k převodu zpráv: "receiver makes good"

Typickým přístupem v návrhu aplikace IBM MQ pro převod kódu je "receiver makes good". Volba "Receiver makes good" snižuje počet převodů zpráv. Vyhýbá se také problému neočekávaných chyb kanálu, pokud dojde k selhání převodu zpráv na některém zprostředkujícím správci front během přenosu zprávy. Pravidlo "receiver makes good" je poškozeno pouze v případě, že existuje nějaký důvod, proč nemůže přijímač provádět dobré zpracování. Přijímající platforma možná nemá správnou znakovou sadu, například.

"Receiver makes good" je také dobré obecné pokyny pro klientské aplikace JMS . Avšak v určitých případech může být konverze na správnou znakovou sadu ve zdroji efektivnější. Konverze z interní reprezentace prostředí JVM musí být provedena, když je odeslána zpráva obsahující text nebo numerické typy. Převod na znakovou sadu požadovanou příjemcem, pokud příjemce není klientem produktu JMS , může odstranit potřebu provedení konverze pro jiného nežJMS příjemce. Je-li příjemce klientem produktu JMS , bude přesto znovu převeden, aby dekódoval data zprávy a vytvořil primitiva a objekty produktu Java .

Rozdíl mezi klientskými aplikacemi JMS a aplikacemi napsanými v jazyku jako C, je, že Java musí provést konverzi dat. Aplikace Java musí převádět čísla a text ze své interní reprezentace do kódovaného formátu používaného ve zprávách.

Nastavením místa určení nebo vlastností zprávy můžete nastavit znakovou sadu a kódování používané produktem IBM MQ ke kódování čísel a textu ve zprávách. Za normálních okolností byste vynechal znakovou sadu jako `1208` a zakódoval jako `Native`.

IBM MQ nekonvertuje bajtová pole. Chcete-li kódovat řetězce a znaková pole do bajtových polí, použijte balík produktu `java.nio.charset` . Parametr `Charset` určuje znakovou sadu použitou k převodu řetězce nebo znakového pole na bajtové pole. bajtové pole lze také dekódovat do řetězce nebo znakového pole pomocí `Charset`. Není dobrým zvykem spoléhat se na `java.nio.charset.Charset.defaultCodePage` při kódování řetězců a znakových polí. Předvolba `Charset` je obvykle `windows-1252` na Windowsa `UTF-8` na UNIX. `windows-1252` je jednobajtová znaková sada a `UTF-8` je vícebajtová znaková sada.

Obecně nechte cílovou znakovou sadu a vlastnosti kódování při jejich výchozích hodnotách `UTF-8` a `Native` při výměně zpráv s jinými aplikacemi produktu JMS . Pokud si vyměňujete zprávy obsahující čísla nebo text s aplikací produktu JMS , vyberte jeden z typů zpráv `JMSTextMessage`, `JMSStreamMessage`, `JMSMapMessage`nebo `JMSObjectMessage` , které odpovídají vašemu účelu. Neexistují žádné další úlohy převodu, které by bylo možné provést.

Pokud si vyměňujete zprávy s aplikacemi jiného typu než typuJMS , které používají formát záznamů, je to složitější. Pokud celý záznam neobsahuje text a může být přenesen jako `JMSTextMessage`, je třeba zakódovat a dekódovat text v aplikaci. Nastavte typ cílové zprávy na `MQ`a použijte `JMSBytesMessage` , aby se zabránilo IBM MQ classes for JMS přidání dalšího záhlaví a označení informací na data zprávy. Použijte metody `JMSBytesMessage` k zápisu čísel a bajtů, a třída `Charset` převádí text do bajtových polí explicitně. Výběr znakové sady může ovlivnit řada faktorů:

- Výkon: Můžete snížit počet převodů transformací textu do znakové sady, která se používá na největším počtu serverů?
- Jednotnost: Přenos všech zpráv ve stejné znakové sadě.
- Richness: Jaké znakové sady mají všechny kódové body, které aplikace musí používat?
- Jednoduchost: jednobajtové znakové sady jsou jednodušší k použití než proměnné délky a vícebajtových znakových sad.

Viz “Výměna formátovaného záznamu s aplikací jinou nežJMS” na stránce 163. Příklady převádění zpráv vyměněných s aplikacemi mimo produktJMS .

Příklady

Tabulka typů zpráv a typů převodu

Tabulka 30. Typy zpráv a typy převodu

	Typ převodu			
Typ zprávy	**Text**	**Číselné**	**Jiný**	**Není**
`JMSObjectMessage`				`getObject` `setObject`
`JMSTextMessage`	`getText` `setText`			

Tabulka 30. Typy zpráv a typy převodu (pokračování)

	Typ převodu			
Typ zprávy	**Text**	**Číselné**	**Jiný**	**Není**
JMSBytesMessage	readUTF writeUTF	readDouble readFloat readInt readLong readShort readUnsignedShort writeDouble writeFloat writeInt writeLong writeShort	readBoolean readObject writeBoolean writeObject	readByte readUnsignedByte readBytes readChar writeByte writeBytes writeChar
JMSStreamMessage	readString writeString	readDouble readFloat readInt readLong readShort writeDouble writeFloat writeInt writeLong writeShort	readBoolean readObject writeBoolean writeObject	readByte readBytes readChar writeByte writeBytes writeChar
JMSMapMessage	getString setString	getDouble getFloat getInt getLong getShort setDouble setFloat setInt setLong setShort	getBoolean getObject setBoolean setObject	getByte getBytes readChar setByte setBytes setChar

```
gmo.Options = MQGMO_WAIT         /* wait for new messages           */
            | MQGMO_NO_SYNCPOINT /* no transaction                  */
            | MQGMO_CONVERT;     /* convert if necessary            */

     while (CompCode != MQCC_FAILED) {
     buflen = sizeof(buffer) - 1; /* buffer size available for GET   */
     memcpy(md.MsgId, MQMI_NONE, sizeof(md.MsgId));
     memcpy(md.CorrelId, MQCI_NONE, sizeof(md.CorrelId));
     md.Encoding       = MQENC_NATIVE;
     md.CodedCharSetId = MQCCSI_Q_MGR;

           MQGET(Hcon,          /* connection handle               */
           Hobj,                /* object handle                   */
           &md,                 /* message descriptor              */
           &gmo,                /* get message options             */
           buflen,              /* buffer length                   */
           buffer,              /* message buffer                  */
           &messlen,            /* message length                  */
           &CompCode,           /* completion code                 */
           &Reason);            /* reason code                     */
```

Obrázek 13. Úsek kódu z amqsget0.c

Odesílání a příjem textu v JMSBytesMessage

Kód v produktu Obrázek 14 na stránce 148 odesílá řetězec do pole BytesMessage. Pro zjednodušení příklad odesílá jeden řetězec, pro který je vhodnější JMSTextMessage . Chcete-li přijmout textový řetězec v bajtová zprávě obsahující směs typů, musíte znát délku řetězce v bajtech, kterému se říká *TEXT_LENGTH* v Obrázek 15 na stránce 148. I v případě řetězce s pevným počtem znaků může být délka znázornění bajtů delší.

```
BytesMessage bytes = session.createBytesMessage();
String codePage = CCSID.getCodepage(((MQDestination) destination)
                        .getIntProperty(WMQConstants.WMQ_CCSID));
bytes.writeBytes("In the destination code page".getBytes(codePage));
producer.send(bytes);
```

Obrázek 14. Odeslání String v JMSBytesMessage

```
BytesMessage message = (BytesMessage)consumer.receive();
int TEXT_LENGTH = new Long(message.getBodyLength())).intValue();
byte[] textBytes = new byte[TEXT_LENGTH];
message.readBytes(textBytes, TEXT_LENGTH);
String codePage = message.getStringProperty(WMQConstants.JMS_IBM_CHARACTER_SET);
String textString = new String(textBytes, codePage);
```

Obrázek 15. Příjem String z JMSBytesMessage

Související pojmy

Převod a kódování zpráv klienta JMS
Jsou zde uvedeny metody, které se používají k provedení konverze a kódování zpráv klienta JMS , s příklady kódu jednotlivých typů konverze.

Převod dat správce front
Převod dat správce front byl vždy k dispozici pro aplikace typu non-JMS , které přijímají zprávy od klientů JMS . Od verze 7.0používají klienti produktu JMS , kteří přijímají zprávy, také převod dat správce front. Počínaje verzí 7.0.1.5nebo 7.0.1.4 s opravou APAR IC72897je konverze dat správce front volitelná.

Související úlohy

Výměna formátovaného záznamu s aplikací jinou nežJMS
Postupujte podle kroků doporučených v této úloze pro návrh a sestavení uživatelské procedury pro převod dat a aplikaci klienta produktu JMS , která může vyměňovat zprávy s aplikací jinou nežJMS pomocí produktu JMSBytesMessage. Výměnu formátované zprávy s aplikací jinou nežJMS se může provést s voláním ukončení konverze dat nebo bez volání ukončení konverze dat.

Související odkazy

Typy a konverze zpráv produktu JMS
Výběr typu zprávy ovlivňuje váš přístup k převodu zpráv. Interakce zprávy o převodu zpráv a typu zprávy jsou popsány pro typy zpráv JMS , JMSObjectMessage, JMSTextMessage, JMSMapMessage, JMSStreamMessagea JMSBytesMessage.

Typy a konverze zpráv produktu JMS
Výběr typu zprávy ovlivňuje váš přístup k převodu zpráv. Interakce zprávy o převodu zpráv a typu zprávy jsou popsány pro typy zpráv JMS , JMSObjectMessage, JMSTextMessage, JMSMapMessage, JMSStreamMessagea JMSBytesMessage.

JMSObjectMessage

Produkt JMSObjectMessage obsahuje jeden objekt a všechny objekty, na které odkazuje, které jsou serializovány do bajtového proudu prostředím JVM. Text je serializován do UTF-8a omezen na řetězce nebo pole znaků ne více než 65534 bajtů. Výhodou produktu JMSObjectMessage je, že aplikace nejsou zapojeny do žádných problémů s převodem dat, dokud budou používat pouze metody a atributy objektu. Produkt JMSObjectMessage poskytuje převod dat pro složité objekty bez programátora aplikace s ohledem na to, jak zakódovat objekt ve zprávě. Nevýhodou použití produktu JMSObjectMessage je možnost výměny dat pouze s jinými aplikacemi produktu JMS . Vyberete-li jeden z dalších typů zpráv produktu JMS , je možné vyměňovat zprávy JMS s aplikacemi mimo produktJMS .

“Odesílání a příjem JMSObjectMessage” na stránce 151 zobrazuje objekt String , který je vyměněn ve zprávě.

Klientská aplikace JMS může přijmout JMSObjectMessage pouze ve zprávě, která má tělo ve stylu JMS. Místo určení musí určovat tělo stylu JMS .

JMSTextMessage

JMSTextMessage obsahuje jeden textový řetězec. Je-li odeslána textová zpráva, je text Format nastaven na "MQSTR ", WMQConstants.MQFMT_STRING. CodedCharacterSetId textu je nastaven na identifikátor kódované znakové sady definovaný pro místo určení. Text je zakódován do CodedCharacterSetId pomocí IBM MQ. Pole CodedCharacterSetId a Format jsou buď nastavena v deskriptoru zpráv, MQMD, nebo do polí JMS v MQRFH2. Pokud je zpráva definována jako mající styl textu zprávy WMQ_MESSAGE_BODY_MQ nebo není zadán styl textu, ale cílové místo určení je WMQ_TARGET_DEST_MQ, pak jsou nastavena pole deskriptoru zpráv. Jinak má zpráva JMS RFH2 a pole jsou nastavena v pevné části MQRFH2.

Aplikace může přepsat identifikátor kódované znakové sady definovaný pro místo určení. Je třeba nastavit vlastnost zprávy JMS_IBM_CHARACTER_SET na identifikátor kódované znakové sady; viz příklad v tématu “Odesílání a příjem JMSTextmessage” na stránce 152.

Když klient JMS zavolá převod správce front metody consumer.receive , je převod volitelný. Převod správce front je povolen nastavením vlastnosti místa určení WMQ_RECEIVE_CONVERSION na hodnotu WMQ_RECEIVE_CONVERSION_QMGR. Správce front převede textovou zprávu z JMS_IBM_CHARACTER_SET určeného pro zprávu před přenosem zprávy na klienta JMS . Znaková sada převedené zprávy je 1208, UTF-8, pokud nemá cíl odlišný WMQ_RECEIVE_CCSID. CodedCharacterSetId ve zprávě, která se odkazuje na JMSTextMessage , je aktualizováno na ID cílové znakové sady. Text je dekódován z cílové znakové sady do Unicode pomocí metody getText ; viz příklad v “Odesílání a příjem JMSTextmessage” na stránce 152.

JMSTextMessage může být odeslán v těle zprávy ve stylu MQbez záhlaví JMS MQRFH2 . Hodnota atributů cíle, WMQ_MESSAGE_BODY a WMQ_TARGET_DEST určují styl těla zprávy,

pokud není potlačeno aplikací. Aplikace může přepsat hodnoty nastavené v místě určení voláním `destination.setMessageBodyStyle(WMQConstants.WMQ_MESSAGE_BODY_MQ)` nebo `destination.setTargetClient(WMQConstants.WMQ_TARGET_DEST_MQ)`.

Pokud odešlete produkt `JMSTextMessage` s tělem stylu MQ odesláním do cíle s hodnotou `WMQ_MESSAGE_BODY` nastavenou na hodnotu `WMQ_MESSAGE_BODY_MQ`, nelze ji přijmout jako `JMSTextMessage` ze stejného cíle. Všechny zprávy přijaté z cíle `WMQ_MESSAGE_BODY` nastavené na hodnotu `WMQ_MESSAGE_BODY_MQ` jsou přijaty jako `JMSBytesMessage`. Pokud se pokusíte přijmout zprávu jako `JMSTextMessage` , způsobí to výjimku, `ClassCastException: com.ibm.jms.JMSBytesMessage cannot be cast to javax.jms.TextMessage`.

Poznámka: Text v `JMSBytesMessage` není převáděn klientem JMS. Klient může přijmout text ve zprávě pouze jako bajtové pole. Je-li povolena konverze správce front, je tento text převeden správcem front, ale klient JMS jej musí stále přijímat jako bajtové pole v `JMSBytesMessage`.

Obecně je lepší použít vlastnost `WMQ_TARGET_DEST` k řízení, zda je produkt `JMSTextMessage` odeslán s použitím stylu těla produktu MQ nebo JMS. Poté můžete zprávu přijmout z cíle, který má buď hodnotu `WMQ_TARGET_DEST` nastavenou na hodnotu `WMQ_TARGET_DEST_MQ` nebo `WMQ_TARGET_DEST_JMS`. `WMQ_TARGET_DEST` nemá žádný vliv na přijímač.

JMSMapMessage a JMSStreamMessage

Tyto dva typy zpráv JMS jsou podobné. Primitivní typy můžete číst a zapisovat do zpráv pomocí metod založených na rozhraních `DataInputStream` a `DataOutputStream` , viz “Tabulka typů zpráv a typů převodu” na stránce 154. Podrobnosti jsou popsány v tématu “Převod a kódování zpráv klienta JMS” na stránce 155. Označené primitivum je označeno; viz “Tělo zprávy produktu JMS” na stránce 140.

Číselná data jsou čtena a zapisována do zprávy kódované jako text XML. Neexistuje žádný odkaz na vlastnost místa určení, `JMS_IBM_ENCODING`. Textová data se zpracovaná stejným způsobem jako text v souboru `JMSTextMessage`. Pokud jste se měli podívat na obsah zprávy vytvořený příkladem v produktu Obrázek 20 na stránce 152, všechna data zprávy by byla ve formátu EBCDIC, protože byla odeslána s hodnotou znakové sady 37.

V produktu `JMSMapMessage` nebo `JMSStreamMessage`můžete odeslat více položek.

Jednotlivé položky dat můžete načíst podle názvu z `JMSMapMessage`, nebo podle pozice z `JMSStreamMessage`. Každá položka je dekódována, když je volána metoda get nebo read za použití hodnoty `CodedCharacterSetId` uložené ve zprávě. Pokud metoda použitá k načtení položky vrací jiný typ na typ, který byl odeslán, typ se převede. Pokud typ nelze převést, dojde k výjimce. Podrobnosti naleznete v tématu Třída JMSStreamMessage . Příklad v příkladu “Odesílání dat v JMSStreamMessage a JMSMapMessage” na stránce 152 ilustruje konverzi typu a získávání obsahu `JMSMapMessage` mimo pořadí.

Pole `MQRFH2.format` pro `JMSMapMessage` a `JMSStreamMessage` je nastaveno na "`MQSTR  `". Je-li vlastnost cíle `WMQ_RECEIVE_CONVERSION` nastavena na hodnotu `WMQ_RECEIVE_CONVERSION_QMGR`, data zprávy se před odesláním klientovi produktu JMS změní na správce front. `MQRFH2.CodedCharacterSetId` zprávy je `WMQ_RECEIVE_CCSID` cíle. `MQRFH2.Encoding` je `Native`. Má-li proměnná `WMQ_RECEIVE_CONVERSION` hodnotu `WMQ_RECEIVE_CONVERSION_CLIENT_MSG` , hodnota `CodedCharacterSetId` a `Encoding` hodnoty `MQRFH2` je hodnota nastavená odesílatelem.

Klientská aplikace JMS může přijmout `JMSMapMessage` nebo `JMSStreamMessage` pouze ve zprávě, která má tělo ve stylu JMSa z místa určení, které nespecifikuje tělo stylu MQ .

JMSBytesMessage

`JMSBytesMessage` může obsahovat více primitivních typů. Primitivní typy můžete číst a zapisovat do zpráv pomocí metod založených na rozhraních `DataInputStream` a `DataOutputStream` , viz “Tabulka typů zpráv a typů převodu” na stránce 154. Podrobnosti jsou popsány v tématu “Typy a konverze zpráv produktu JMS” na stránce 149.

Kódování číselných dat ve zprávě je určováno hodnotou `JMS_IBM_ENCODING` , která je nastavena před zápisem číselných dat do `JMSBytesMessage`. Aplikace může přepsat výchozí kódování `Native` definované pro `JMSBytesMessage` nastavením vlastnosti zprávy `JMS_IBM_ENCODING`.

Textová data lze číst a zapisovat do `UTF-8` pomocí `readUTF` a `writeUTF`, nebo v Unicode pomocí metod `readChar` a `writeChar` . K dispozici nejsou žádné metody, které by používaly `CodedCharacterSetId`. Alternativně může klient JMS kódovat a dekódovat text do bajtů pomocí třídy `Charset` . Přenese bajty mezi prostředím JVM a zprávou bez provedení převodu IBM MQ classes for JMS ; viz “Odesílání a příjem textu v JMSBytesMessage” na stránce 152.

`JMSBytesMessage` odeslané do aplikace MQ se obvykle odesílá do těla zprávy ve stylu MQbez záhlaví JMS `MQRFH2` . Je-li odeslán aplikaci produktu JMS , styl těla zprávy je obvykle JMS. Hodnota atributů cíle, `WMQ_MESSAGE_BODY` a `WMQ_TARGET_DEST` určují styl těla zprávy, pokud není potlačeno aplikací. Aplikace může přepsat hodnoty nastavené v místě určení voláním `destination.setMessageBodyStyle(WMQConstants.WMQ_MESSAGE_BODY_MQ)` nebo `destination.setTargetClient(WMQConstants.WMQ_TARGET_DEST_MQ)`.

Pokud odešlete `JMSBytesMessage` s tělem stylu MQ , můžete přijmout zprávu z cíle, který definuje buď styl těla zprávy produktu MQ nebo zprávy produktu JMS . Pokud odešlete `JMSBytesMessage` s tělem stylu JMS, musíte přijmout zprávu z cíle, který definuje styl textu zprávy produktu JMS . Pokud ji neuděláte, bude se s produktem `MQRFH2` zacházet jako s částí dat zprávy uživatele, což nemusí být to, co očekáváte.

Určuje, zda má zpráva styl těla produktu MQ nebo JMS , tím, jak je doručen, není ovlivněn nastavením hodnoty `WMQ_TARGET_DEST`.

Zpráva může být později transformována správcem front, pokud je pro data zprávy poskytnuta služba `Format` a je povolen převod dat správce front. Nepoužívejte pole formátu pro cokoli jiného než uvedení formátu dat zprávy, nebo ponechte prázdné, `MQConstants.MQFMT_NONE`

V produktu `JMSBytesMessage`můžete odeslat více položek. Každá číselná položka se převede, když je zpráva odeslána s použitím kódování definovaného pro zprávu.

Jednotlivé položky dat můžete načíst z produktu `JMSBytesMessage`. Volejte metody čtení ve stejném pořadí, v jakém byly volány metody zápisu pro vytvoření zprávy. Každá číselná položka se převede, když se volá zpráva pomocí hodnoty `Encoding` uložené ve zprávě.

Na rozdíl od `JMSMapMessage` a `JMSStreamMessage`, `JMSBytesMessage` obsahuje pouze data zapsaná aplikací. Do dat zprávy nejsou uložena žádná další data, jako např. značky XML použité k definování položek v produktu `JMSMapMessage` a `JMSStreamMessage`. Z tohoto důvodu použijte produkt `JMSBytesMessage` k přenosu zpráv naformátovaných pro jiné aplikace.

Převod mezi `JMSBytesMessage` a `DataInputStream` a `DataOutputStream` je užitečný v některých aplikacích. Kód založený na příkladu, “Čtení a zápis zpráv pomocí DataInputStream a DataOutputStream” na stránce 153, je nezbytný pro použití balíku `com.ibm.mq.header` s JMS.

Příklady

Odesílání a příjem `JMSObjectMessage`

```
ObjectMessage omo = session.createObjectMessage();
omo.setObject(new String("A string"));
producer.send(omo);
...
ObjectMessage omi = (ObjectMessage)consumer.receive();
System.out.println((String)omi.getObject());
...
A string
```

Obrázek 16. Odesílání a příjem `JMSObjectMessage`

Odesílání a příjem JMSTextmessage

Textová zpráva nemůže obsahovat text v různých znakových sadách. Příklad ukazuje text v různých znakových sadách, který je odeslán ve dvou různých zprávách.

```
TextMessage tmo = session.createTextMessage();
tmo.setText("Sent in the character set defined for the destination");
producer.send(tmo);
```

Obrázek 17. Odeslat textovou zprávu ve znakové sadě definované v místě určení

```
TextMessage tmo = session.createTextMessage();
tmo.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET, 37);
tmo.setText("Sent in EBCDIC character set 37");
producer.send(tmo);
```

Obrázek 18. Odeslat textovou zprávu v produktu `ccsid 37`

```
TextMessage tmi = (TextMessage)consumer.receive();
System.out.println(tmi.getText());
...
Sent in the character set defined for the destination
```

Obrázek 19. Přijmout textovou zprávu

Odesílání dat v JMSStreamMessage a JMSMapMessage

```
StreamMessage smo = session.createStreamMessage();
smo.writeString("256");
smo.writeInt(512);
smo.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET, 37);
producer.send(smo);
...
MapMessage mmo = session.createMapMessage();
mmo.setString("First", "256");
mmo.setInt("Second", 512);
mmo.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET, 37);
producer.send(mmo);
...
StreamMessage smi = (StreamMessage)consumer.receive();
System.out.println("Stream: First as float "   + smi.readFloat() +
                           " Second as String " + smi.readString());
...
Stream: First as float: 256.0, Second as String: 512
...
MapMessage mmi = (MapMessage)consumer.receive();
System.out.println("Map: Second as String " + mmi.getString("Second") +
                         " First as double "  + mmi.getDouble("First"));
...
Map: Second as String: 512, First as double: 256.0
```

Obrázek 20. Odeslat data v produktu JMS`StreamMessage` a JMS`MapMessage`

Odesílání a příjem textu v JMSBytesMessage

Kód v produktu Obrázek 21 na stránce 153 odesílá řetězec do pole BytesMessage. Pro zjednodušení příklad odesílá jeden řetězec, pro který je vhodnější JMS`TextMessage` . Chcete-li přijmout textový

řetězec v bajtová zprávě obsahující směs typů, musíte znát délku řetězce v bajtech, kterému se říká *TEXT_LENGTH* v Obrázek 22 na stránce 153. I v případě řetězce s pevným počtem znaků může být délka znázornění bajtů delší.

```
BytesMessage bytes = session.createBytesMessage();
String codePage = CCSID.getCodepage(((MQDestination) destination)
                        .getIntProperty(WMQConstants.WMQ_CCSID));
bytes.writeBytes("In the destination code page".getBytes(codePage));
producer.send(bytes);
```

Obrázek 21. Odeslání `String` v `JMSBytesMessage`

```
BytesMessage message = (BytesMessage)consumer.receive();
int TEXT_LENGTH = new Long(message.getBodyLength())).intValue();
byte[] textBytes = new byte[TEXT_LENGTH];
message.readBytes(textBytes, TEXT_LENGTH);
String codePage = message.getStringProperty(WMQConstants.JMS_IBM_CHARACTER_SET);
String textString = new String(textBytes, codePage);
```

Obrázek 22. Příjem `String` z `JMSBytesMessage`

Čtení a zápis zpráv pomocí `DataInputStream` a `DataOutputStream`

Kód v produktu Obrázek 23 na stránce 153 vytváří `JMSBytesMessage` pomocí `DataOutputStream`.

```
ByteArrayOutputStream bout = new ByteArrayOutputStream();
DataOutputStream dout = new DataOutputStream(bout);
BytesMessage messageOut = prod.session.createBytesMessage();
// messageOut.setIntProperty(WMQConstants.JMS_IBM_ENCODING,
//                           ((MQDestination) (prod.destination)).getIntProperty
//                           (WMQConstants.WMQ_ENCODING));
int ccsidOut = (((MQDestination)prod.destination).getIntProperty(WMQConstants.WMQ_CCSID));
String codePageOut = CCSID.getCodepage(ccsidOut);
dout.writeInt(ccsidOut);
dout.write(codePageOut.getBytes(codePageOut));
messageOut.writeBytes(bout.toByteArray());
producer.send(messageOut);
```

Obrázek 23. Odeslat `JMSBytesMessage` pomocí `DataOutputStream`

Příkaz, který nastaví vlastnost `JMS_IBM_ENCODING` , je označen jako komentář. Příkaz je platný, pokud je zapisoval přímo do `JMSBytesMessage`, ale nemá žádný účinek při zápisu do `DataOutputStream`. Čísla, která jsou zapsána do produktu `DataOutputStream` , jsou zakódována v kódování `Native` . Nastavení `JMS_IBM_ENCODING` nemá žádný efekt.

Kód v produktu Obrázek 24 na stránce 154 přijímá `JMSBytesMessage` pomocí `DataInputStream`.

```
static final int ccsidIn_SIZE = (Integer.SIZE)/8;
...
connection.start();
BytesMessage messageIn = (BytesMessage) consumer.receive();
int messageLength = new Long(messageIn.getBodyLength()).intValue();
byte [] bin = new byte[messageLength];
messageIn.readBytes(bin, messageLength);
DataInputStream din = new DataInputStream(new ByteArrayInputStream(bin));
int ccsidIn = din.readInt();
byte [] codePageByte = new byte[messageLength - ccsidIn_SIZE];
din.read(codePageByte, 0, codePageByte.length);
System.out.println("CCSID " + ccsidIn + " code page " + new String(codePageByte,
                    messageIn.getStringProperty(WMQConstants.JMS_IBM_CHARACTER_SET)));
```

Obrázek 24. Přijmout JMSBytesMessage pomocí DataInputStream

Kódová stránka je vytištěna pomocí vlastnosti kódové stránky vstupních dat zprávy, JMS_IBM_CHARACTER_SET. Na vstupu JMS_IBM_CHARACTER_SET je kódová stránka Java a nejedná se o číselný identifikátor kódované znakové sady.

Tabulka typů zpráv a typů převodu

Tabulka 31. Typy zpráv a typy převodu

	Typ převodu			
Typ zprávy	**Text**	**Číselné**	**Jiný**	**Není**
JMSObjectMessage				getObject setObject
JMSTextMessage	getText setText			
JMSBytesMessage	readUTF writeUTF	readDouble readFloat readInt readLong readShort readUnsignedShort writeDouble writeFloat writeInt writeLong writeShort	readBoolean readObject writeBoolean writeObject	readByte readUnsignedByte readBytes readChar writeByte writeBytes writeChar

Tabulka 31. Typy zpráv a typy převodu (pokračování)				
	Typ převodu			
Typ zprávy	**Text**	**Číselné**	**Jiný**	**Není**
JMSStreamMessage	readString writeString	readDouble readFloat readInt readLong readShort writeDouble writeFloat writeInt writeLong writeShort	readBoolean readObject writeBoolean writeObject	readByte readBytes readChar writeByte writeBytes writeChar
JMSMapMessage	getString setString	getDouble getFloat getInt getLong getShort setDouble setFloat setInt setLong setShort	getBoolean getObject setBoolean setObject	getByte getBytes readChar setByte setBytes setChar

Související pojmy

Metody konverze zpráv produktu JMS

Návrháři aplikací produktu JMS jsou otevřeny pro řadu přístupů k převodu dat. Tyto přístupy se nevylučují; některé aplikace budou pravděpodobně používat kombinaci těchto přístupů. Pokud si aplikace vyměňuje pouze text nebo si vyměňuje zprávy pouze s jinými aplikacemi produktu JMS , obvykle nepovažujete převod dat za normální. Převod dat se provádí automaticky pro vás, příkazem IBM MQ.

Převod a kódování zpráv klienta JMS

Jsou zde uvedeny metody, které se používají k provedení konverze a kódování zpráv klienta JMS , s příklady kódu jednotlivých typů konverze.

Převod dat správce front

Převod dat správce front byl vždy k dispozici pro aplikace typu non-JMS , které přijímají zprávy od klientů JMS . Od verze 7.0používají klienti produktu JMS , kteří přijímají zprávy, také převod dat správce front. Počínaje verzí 7.0.1.5nebo 7.0.1.4 s opravou APAR IC72897je konverze dat správce front volitelná.

Související úlohy

Výměna formátovaného záznamu s aplikací jinou nežJMS

Postupujte podle kroků doporučených v této úloze pro návrh a sestavení uživatelské procedury pro převod dat a aplikaci klienta produktu JMS , která může vyměňovat zprávy s aplikací jinou nežJMS pomocí produktu JMSBytesMessage. Výměnu formátované zprávy s aplikací jinou nežJMS se může provést s voláním ukončení konverze dat nebo bez volání ukončení konverze dat.

Převod a kódování zpráv klienta JMS

Jsou zde uvedeny metody, které se používají k provedení konverze a kódování zpráv klienta JMS , s příklady kódu jednotlivých typů konverze.

Převod a kódování se provádí při čtení nebo zápisu primitiv nebo objektů produktu Java do zpráv produktu JMS a jejich zápisu do nich. Převod se nazývá konverze dat klienta JMS , aby se odlišil od konverze dat správce front a konverze aplikačních dat. Konverze probíhá striktně, když se data čtou ze zprávy JMS nebo jsou do ní zapsány. Text je převeden na a z interní 16bitové reprezentace Unicode[3]do znakové sady

použité pro text ve zprávách. Číselná data jsou převedena na a Java primitivní číselné typy na kódování definované pro zprávu. Zda se provádí převod a jaký typ převodu se provede, závisí na typu zprávy JMS a na operaci čtení nebo zápisu.

Tabulka 32 na stránce 156 kategorizuje metody čtení a zápisu pro různé typy zpráv JMS podle typu prováděné konverze. Typy převodů jsou popsány v textu následujícím za tabulkou.

Tabulka 32. Typy zpráv a typy převodu

	Typ převodu			
Typ zprávy	**Text**	**Číselné**	**Jiný**	**Není**
JMSObjectMessage				getObject setObject
JMSTextMessage	getText setText			
JMSBytesMessage	readUTF writeUTF	readDouble readFloat readInt readLong readShort readUnsignedShort writeDouble writeFloat writeInt writeLong writeShort	readBoolean readObject writeBoolean writeObject	readByte readUnsignedByte readBytes readChar writeByte writeBytes writeChar
JMSStreamMessage	readString writeString	readDouble readFloat readInt readLong readShort writeDouble writeFloat writeInt writeLong writeShort	readBoolean readObject writeBoolean writeObject	readByte readBytes readChar writeByte writeBytes writeChar
JMSMapMessage	getString setString	getDouble getFloat getInt getLong getShort setDouble setFloat setInt setLong setShort	getBoolean getObject setBoolean setObject	getByte getBytes readChar setByte setBytes setChar

[3] Některá znázornění Unicode vyžadují více než 16 bitů. Viz odkaz na produkt Java SE.

Text

Standardní hodnota `CodedCharacterSetId` pro místo určení je 1208, `UTF-8`. Standardně je text převeden z Unicode a odeslán jako textový řetězec `UTF-8` . Při přijetí se text převede z kódované znakové sady ve zprávě přijaté klientem do Unicode.

Metody `setText` a `writeString` převádějí text z kódování Unicode do znakové sady definované pro místo určení. Aplikace může přepsat cílovou znakovou sadu nastavením vlastnosti zprávy `JMS_IBM_CHARACTER_SET`. `JMS_IBM_CHARACTER_SET`, při odeslání zprávy musí být číselný identifikátor kódované znakové sady[4].

Úseky kódu v produktu “Odesílání a příjem JMSTextmessage” na stránce 159 posílají dvě zprávy. Jedna se odešle ve znakové sadě definované pro místo určení a druhá ve znakové sadě 37, definované aplikací.

Metody `getText` a `readString` převádějí text ve zprávě ze znakové sady definované ve zprávě do Unicode. Metody používají kódovou stránku definovanou ve vlastnosti zprávy `JMS_IBM_CHARACTER_SET`. Kódová stránka je mapována z produktu `MQRFH2.CodedCharacterSetId` , pokud se nejedná o zprávu typu MQ, která nemá `MQRFH2`. Pokud se jedná o zprávu typu MQtypu -type bez příkazu `MQRFH2`, je kódová stránka mapována z produktu `MQMD.CodedCharacterSetId`.

Úsek kódu v produktu Obrázek 29 na stránce 159 přijímá zprávu, která byla odeslána do cíle. Text ve zprávě je převeden z kódové stránky `IBM037` zpět do Unicode.

Poznámka: Jednoduchým způsobem, jak zkontrolovat, zda je text převeden na kódovanou znakovou sadu 37, je použít Průzkumníka IBM MQ . Procházet frontu a zobrazit vlastnosti zprávy před jeho načtením.

Porovnejte úsek kódu v produktu Obrázek 28 na stránce 159 s chybným úsekem kódu v produktu Obrázek 25 na stránce 157. V chybném úseku je textový řetězec převeden dvakrát, jednou aplikací, a znovu produktem IBM MQ.

```
TextMessage tmo = session.createTextMessage();
tmo.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET, 37);
tmo.setText(new String("Sent in EBCDIC character set 37".getBytes(CCSID.getCodepage(37))));
producer.send(tmo);
```

Obrázek 25. Nesprávný převod kódové stránky

Metoda `writeUTF` převádí text z kódování Unicode na 1208, `UTF-8`. Textový řetězec je uváděn s délkou 2 bajtové délky. Maximální délka textového řetězce je 65534 bajtů. Metoda `readUTF` čte položku ve zprávě napsanou metodou `writeUTF` . Přečte přesně počet bajtů, které byly zapsány metodou `writeUTF` .

Číselné

Výchozí číselné kódování pro místo určení je `Native`. Kódovací konstanta `Native` pro Java má hodnotu 273, `x'00000111'`, což je stejné pro všechny platformy. Při přijetí jsou čísla ve zprávě správně transformována do číselných primitiv Java . Transformace používá kódování definované ve zprávě a typ vrácený metodou čtení.

Metoda odeslání převádí čísla, která jsou přidána do zprávy `set` a `write` , do číselného kódování definovaného pro místo určení. Kódování cíle lze přepsat pro zprávu aplikací nastavující vlastnost zprávy `JMS_IBM_ENCODING` ; například:

```
message.setIntProperty(WMQConstants.JMS_IBM_ENCODING,
WMQConstants.WMQ_ENCODING_INTEGER_REVERSED);
```

[4] Při přijetí zprávy `JMS_IBM_CHARACTER_SET` je název kódové stránky Java `Charset` .

Numerické metody `get` a `read` převádějí čísla ve zprávě z numerického kódování definovaného ve zprávě. Přemění čísla na typ, který je určen metodou `read` nebo `get` ; viz vlastnost ENCODING. Metody používají kódování definované v produktu `JMS_IBM_ENCODING`. Kódování je mapováno z `MQRFH2.Encoding` , pokud se nejedná o zprávu typu MQ, která nemá `MQRFH2`. Pokud se jedná o zprávu typu MQtypu -type bez příkazu `MQRFH2`, pak metody používají kódování definované v produktu `MQMD.Encoding`.

Příklad v produktu Obrázek 30 na stránce 160 zobrazuje aplikaci kódování čísla v cílovém formátu a odeslání je v souboru `JMSStreamMessage`. Porovnejte příklad v produktu Obrázek 30 na stránce 160 s příkladem v produktu Obrázek 31 na stránce 160. Rozdíl je v tom, že `JMS_IBM_ENCODING` musí být nastaveno v `JMSBytesMessage`.

Poznámka: Jednoduchým způsobem, jak zkontrolovat, zda je číslo zakódováno správně, je použití Průzkumníka IBM MQ . Procházejte frontu a zobrazte vlastnosti zprávy předtím, než je spotřebována.

Jiný

Metody `boolean` zakóduje `true` a `false` jako `x'01'` a `x'00'` v `JMSByteMessage`, `JMSStreamMessage`a `JMSMapMessage`.

Metody UTF zakóduje a dekóduje Unicode do textových řetězců `UTF-8` . Řetězce jsou omezeny na méně než 65536 znaků a jsou jim uvedeny 2 bajtové pole délky.

Primitivní typy objektů jsou zalamování objektů jako objekty. Číselné a textové typy jsou kódovány nebo převedeny, jako kdyby byly primitivní typy přečteny nebo zapsány pomocí číselných a textových metod.

Není

Metody `readByte`, `readBytes`, `readUnsignedByte`, `writeByte`a `writeBytes` získají nebo vloží jednotlivé bajty nebo pole bajtů mezi aplikací a zprávou bez konverze. Metody `readChar` a `writeChar` mohou používat a vkládat 2 bajtové znaky Unicode mezi aplikací a zprávou bez konverze.

Pomocí metod `readBytes` a `writeBytes` může aplikace provádět svůj vlastní převod kódových bodů stejně jako v produktu “Odesílání a příjem textu v JMSBytesMessage” na stránce 160.

Produkt IBM MQ neprovedl v klientovi žádnou konverzi kódové stránky, protože zpráva je `JMSBytesMessage`, a protože jsou použity metody `readBytes` a `writeBytes` . Nicméně, pokud bajty představují text, ujistěte se, že kódová stránka použitá aplikací odpovídá kódované znakové sadě cíle. Zpráva může být znovu převedena pomocí uživatelské procedury pro převod správce front. Jinou možností je to, že přijímající klientský program JMS může podle konvence konvertovat libovolná bajtová pole reprezentující text ve zprávě do řetězců nebo znaků pomocí vlastnosti `JMS_IBM_CHARACTER_SET` ve zprávě.

V tomto příkladu klient používá cílovou kódovanou znakovou sadu pro svůj převod:

```
bytes.writeBytes("In the destination code page".getBytes(
CCSID.getCodepage(((MQDestination) destination)
.getIntProperty(WMQConstants.WMQ_CCSID))));
```

Případně může klient zvolit kódovou stránku a poté nastavit příslušnou kódovanou znakovou sadu ve vlastnosti `JMS_IBM_CHARACTER_SET` dané zprávy. IBM MQ classes for Java používá `JMS_IBM_CHARACTER_SET` k nastavení pole `CodedCharacterSetId` ve vlastnostech JMS v `MQRFH2`, nebo v deskriptoru zprávy, `MQMD`:

```
String codePage = CCSID.getCodepage(37);
message.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET, codePage);
5
```

[5] `SetStringProperty(WMQConstants.JMS_IBM_CHARACTER_SET, codePage)` currently accepts only numeric character set identifiers.

Je-li bajtové pole zapsáno do JMSStringMessage nebo JMSMapMessage, IBM MQ classes for JMS neprovede konverzi dat, protože bajty jsou zapsány jako hexadecimální data, nikoli jako text v JMSStringMessage a JMSMapMessage.

Pokud bajty představují znaky ve vaší aplikaci, musíte vzít v úvahu, jaké kódové body chcete číst a zapisovat do zprávy. Kód v souboru Obrázek 26 na stránce 159 se řídí konvencemi použití cílové kódové znakové sady. Pokud vytvoříte řetězec pomocí výchozí znakové sady pro prostředí JVM, bude obsah bajtů záviset na platformě. Prostředí JVM v systému Windows má obvykle standardní hodnotu Charset z windows-1252a UNIX, UTF-8. Výměna mezi Windows a UNIX vyžaduje, abyste vybrali explicitní kódovou stránku pro výměnu textu jako bajty.

```
StreamMessage smo = producer.session.createStreamMessage();
smo.writeBytes("123".getBytes(CCSID.getCodepage(((MQDestination) destination)
.getIntProperty(WMQConstants.WMQ_CCSID))));
```

Obrázek 26. Psaní bajtů představujících řetězec v JMSStreamMessage s použitím cílové znakové sady

Příklady

Odesílání a příjem JMSTextmessage

Textová zpráva nemůže obsahovat text v různých znakových sadách. Příklad ukazuje text v různých znakových sadách, který je odeslán ve dvou různých zprávách.

```
TextMessage tmo = session.createTextMessage();
tmo.setText("Sent in the character set defined for the destination");
producer.send(tmo);
```

Obrázek 27. Odeslat textovou zprávu ve znakové sadě definované v místě určení

```
TextMessage tmo = session.createTextMessage();
tmo.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET, 37);
tmo.setText("Sent in EBCDIC character set 37");
producer.send(tmo);
```

Obrázek 28. Odeslat textovou zprávu v produktu ccsid 37

```
TextMessage tmi = (TextMessage)consumer.receive();
System.out.println(tmi.getText());
...
Sent in the character set defined for the destination
```

Obrázek 29. Přijmout textovou zprávu

Příklady kódování

Příklady, které ukazují číslo odesílané v kódování, definuje místo určení. Všimněte si, že je třeba nastavit vlastnost JMS_IBM_ENCODING hodnoty JMSBytesMessage na hodnotu zadanou pro místo určení.

```
StreamMessage smo = session.createStreamMessage();
smo.writeInt(256);
producer.send(smo);
...
StreamMessage smi = (StreamMessage)consumer.receive();
System.out.println(smi.readInt());
...
256
```

Obrázek 30. Odesílání čísla s použitím kódování cíle v produktu JMS`StreamMessage`

```
BytesMessage bmo = session.createBytesMessage();
bmo.writeInt(256);
int encoding = ((MQDestination) (destination)).getIntProperty
    (WMQConstants.WMQ_ENCODING)
bmo.setIntProperty(WMQConstants.JMS_IBM_ENCODING, encoding);
producer.send(bmo);
...
BytesMessage bmi = (BytesMessage)consumer.receive();
System.out.println(bmi.readInt());
...
256
```

Obrázek 31. Odesílání čísla s použitím kódování cíle v produktu JMS`BytesMessage`

Odesílání a příjem textu v JMSBytesMessage

Kód v produktu Obrázek 32 na stránce 160 odesílá řetězec do pole BytesMessage. Pro zjednodušení příklad odesílá jeden řetězec, pro který je vhodnější JMS`TextMessage` . Chcete-li přijmout textový řetězec v bajtová zprávě obsahující směs typů, musíte znát délku řetězce v bajtech, kterému se říká *TEXT_LENGTH* v Obrázek 33 na stránce 160. I v případě řetězce s pevným počtem znaků může být délka znázornění bajtů delší.

```
BytesMessage bytes = session.createBytesMessage();
String codePage = CCSID.getCodepage(((MQDestination) destination)
                  .getIntProperty(WMQConstants.WMQ_CCSID));
bytes.writeBytes("In the destination code page".getBytes(codePage));
producer.send(bytes);
```

Obrázek 32. Odeslání `String` v `JMSBytesMessage`

```
BytesMessage message = (BytesMessage)consumer.receive();
int TEXT_LENGTH = new Long(message.getBodyLength())).intValue();
byte[] textBytes = new byte[TEXT_LENGTH];
message.readBytes(textBytes, TEXT_LENGTH);
String codePage = message.getStringProperty(WMQConstants.JMS_IBM_CHARACTER_SET);
String textString = new String(textBytes, codePage);
```

Obrázek 33. Příjem `String` z `JMSBytesMessage`

Související pojmy

Metody konverze zpráv produktu JMS

Návrháři aplikací produktu JMS jsou otevřeny pro řadu přístupů k převodu dat. Tyto přístupy se nevylučují; některé aplikace budou pravděpodobně používat kombinaci těchto přístupů. Pokud si aplikace vyměňuje

pouze text nebo si vyměňuje zprávy pouze s jinými aplikacemi produktu JMS , obvykle nepovažujete převod dat za normální. Převod dat se provádí automaticky pro vás, příkazem IBM MQ.

Převod dat správce front
Převod dat správce front byl vždy k dispozici pro aplikace typu non-JMS , které přijímají zprávy od klientů JMS . Od verze 7.0používají klienti produktu JMS , kteří přijímají zprávy, také převod dat správce front. Počínaje verzí 7.0.1.5nebo 7.0.1.4 s opravou APAR IC72897je konverze dat správce front volitelná.

Související úlohy

Výměna formátovaného záznamu s aplikací jinou nežJMS
Postupujte podle kroků doporučených v této úloze pro návrh a sestavení uživatelské procedury pro převod dat a aplikaci klienta produktu JMS , která může vyměňovat zprávy s aplikací jinou nežJMS pomocí produktu `JMSBytesMessage`. Výměnu formátované zprávy s aplikací jinou nežJMS se může provést s voláním ukončení konverze dat nebo bez volání ukončení konverze dat.

Související odkazy

Typy a konverze zpráv produktu JMS
Výběr typu zprávy ovlivňuje váš přístup k převodu zpráv. Interakce zprávy o převodu zpráv a typu zprávy jsou popsány pro typy zpráv JMS , `JMSObjectMessage`, `JMSTextMessage`, `JMSMapMessage`, `JMSStreamMessage`a `JMSBytesMessage`.

Převod dat správce front
Převod dat správce front byl vždy k dispozici pro aplikace typu non-JMS , které přijímají zprávy od klientů JMS . Od verze 7.0používají klienti produktu JMS , kteří přijímají zprávy, také převod dat správce front. Počínaje verzí 7.0.1.5nebo 7.0.1.4 s opravou APAR IC72897je konverze dat správce front volitelná.

Správce front může převádět znaková data a číselná data v datech zprávy pomocí hodnot `CodedCharacterSetId`, `Encoding`a `Format` nastavených pro data zprávy. Pro aplikace jiné nežJMS je schopnost převodu vždy k dispozici nastavením volby `GetMessageOption`, `GMO_CONVERT`. Schopnost převodu správce front nebyla k dispozici pro aplikaci produktu JMS přijímající zprávu do 7.0.

Převod správce front můžete použít před verzí 7.0s klientskou aplikací JMS , která odesílá zprávu. Klient JMS sestaví formátovaný záznam, nastaví atributy `CodedCharacterSetId`, `Encoding`a `Format` odpovídající datům umístěnými ve zprávě. Přijímací aplikace typu non-JMS přečte zprávu pomocí příkazu `GMO_CONVERT`a způsobí, že bude volána uživatelská procedura pro převod dat uživatele. Uživatelská procedura pro převod dat je sdílená knihovna, která má název nastavený v poli `Format` .

Od verze 7.0je správce front schopen převést zprávy odesílané na klienty produktu JMS . Od verze 7.0.0.0 do verze 7.0.1.4 včetně je pro klienty JMS vždy volán převod správce front. Od verze 7.0.1.5nebo z verze 7.0.1.4 s použitím opravy APAR IC72897 je převod správce front řízen nastavením vlastnosti cíle `WMQ_RECEIVE_CONVERSION`na hodnotu `WMQ_RECEIVE_CONVERSION_QMGR`nebo `WMQ_RECEIVE_CONVERSION_CLIENT_MSG`. `WMQ_RECEIVE_CONVERSION_CLIENT_MSG` je výchozí nastavení, které odpovídá chování produktu IBM WebSphere MQ 6.0, které nepodporuje převod dat správce front pro klienty JMS . Aplikace může změnit nastavení místa určení:

```
((MQDestination)destination).setIntProperty(
              WMQConstants.WMQ_RECEIVE_CONVERSION,
              WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR);
```

Nebo,

```
((MQDestination)destination).setReceiveConversion
              (WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR);
```

Obrázek 34. Povolit převod dat správce front

Převod dat správce front pro klienta JMS se provádí, když klient volá metodu `consumer.receive` . Textová data jsou při výchozím nastavení transformována do formátu UTF-8 (1208). Následné čtení a získání textu dekóduje text v přijatých datech z UTF-8, vytvoří Java textová primitiva ve svém interním

kódování Unicode. UTF-8 není jediná cílová znaková sada z převodu dat správce front. Nastavením vlastnosti cíle WMQ_RECEIVE_CCSID si můžete vybrat jiný CCSID.

Aplikace může také změnit nastavení cíle, například nastavení na 437, DOS-US:

```
((MQDestination)destination).setIntProperty
(WMQConstants.WMQ_RECEIVE_CCSID, 437);
```

Nebo,

```
((MQDestination)destination).setReceiveCCSID(437);
```

Obrázek 35. Nastavit cílovou kódovou sadu znaků pro převod správce front

Důvod změny WMQ_RECEIVE_CCSID je specializovaný; zvolený CCSID nečiní žádný rozdíl pro textové objekty vytvořené v prostředí JVM. Některá prostředí JVM, na některých platformách, však nemusí být schopna zpracovat převod z CCSID textu ve zprávě do Unicode. Tato volba vám dává volbu CCSID pro jakýkoli text doručený klientovi ve zprávě. Některé klientské platformy JMS mají problémy s textem zprávy dodávanými v UTF-8.

Kód JMS je ekvivalentem tučného textu v kódu C v souboru Obrázek 36 na stránce 162,

```
gmo.Options = MQGMO_WAIT         /* wait for new messages           */
            | MQGMO_NO_SYNCPOINT /* no transaction                  */
            | MQGMO_CONVERT;     /* convert if necessary            */

     while (CompCode != MQCC_FAILED) {
     buflen = sizeof(buffer) - 1; /* buffer size available for GET   */
     memcpy(md.MsgId, MQMI_NONE, sizeof(md.MsgId));
     memcpy(md.CorrelId, MQCI_NONE, sizeof(md.CorrelId));
     md.Encoding       = MQENC_NATIVE;
     md.CodedCharSetId = MQCCSI_Q_MGR;

           MQGET(Hcon,          /* connection handle               */
           Hobj,                /* object handle                   */
           &md,                 /* message descriptor              */
           &gmo,                /* get message options             */
           buflen,              /* buffer length                   */
           buffer,              /* message buffer                  */
           &messlen,            /* message length                  */
           &CompCode,           /* completion code                 */
           &Reason);            /* reason code                     */
```

Obrázek 36. Úsek kódu z amqsget0.c

Poznámka:

Převod správce front se provádí pouze na datech zprávy, která mají známý formát IBM MQ . MQSTR, nebo MQCIH jsou příklady známých formátů, které jsou předdefinované. Známý formát může být také uživatelem definovaný formát, pokud jste zadali uživatelskou proceduru pro převod dat.

Zprávy vytvořené jako JMSTextMessage, JMSMapMessage a JMSStreamMessage, mají formát MQSTR a mohou být převedeny správcem front.

Související pojmy

Metody konverze zpráv produktu JMS

Návrháři aplikací produktu JMS jsou otevřeny pro řadu přístupů k převodu dat. Tyto přístupy se nevylučují; některé aplikace budou pravděpodobně používat kombinaci těchto přístupů. Pokud si aplikace vyměňuje pouze text nebo si vyměňuje zprávy pouze s jinými aplikacemi produktu JMS , obvykle nepovažujete převod dat za normální. Převod dat se provádí automaticky pro vás, příkazem IBM MQ.

Převod a kódování zpráv klienta JMS

Jsou zde uvedeny metody, které se používají k provedení konverze a kódování zpráv klienta JMS , s příklady kódu jednotlivých typů konverze.

“Vyvolání uživatelské procedury pro převod dat” na stránce 949
Uživatelská procedura pro převod dat je uživatelská procedura, která přijímá řízení během zpracování volání MQGET.

Související úlohy

Výměna formátovaného záznamu s aplikací jinou nežJMS
Postupujte podle kroků doporučených v této úloze pro návrh a sestavení uživatelské procedury pro převod dat a aplikaci klienta produktu JMS , která může vyměňovat zprávy s aplikací jinou nežJMS pomocí produktu `JMSBytesMessage`. Výměnu formátované zprávy s aplikací jinou nežJMS se může provést s voláním ukončení konverze dat nebo bez volání ukončení konverze dat.

Související odkazy

Typy a konverze zpráv produktu JMS
Výběr typu zprávy ovlivňuje váš přístup k převodu zpráv. Interakce zprávy o převodu zpráv a typu zprávy jsou popsány pro typy zpráv JMS , `JMSObjectMessage`, `JMSTextMessage`, `JMSMapMessage`, `JMSStreamMessage`a `JMSBytesMessage`.

Výměna formátovaného záznamu s aplikací jinou nežJMS
Postupujte podle kroků doporučených v této úloze pro návrh a sestavení uživatelské procedury pro převod dat a aplikaci klienta produktu JMS , která může vyměňovat zprávy s aplikací jinou nežJMS pomocí produktu `JMSBytesMessage`. Výměnu formátované zprávy s aplikací jinou nežJMS se může provést s voláním ukončení konverze dat nebo bez volání ukončení konverze dat.

Než začnete

Můžete být schopni navrhnout jednodušší řešení pro výměnu zpráv s aplikací mimo produktJMS pomocí produktu `JMSTextMessage`. Před provedením kroků uvedených v této úloze tuto možnost eliminujte.

Informace o této úloze

Klient produktu JMS je snazší psát, pokud se nezapojuje do podrobností formátování zpráv JMS vyměněných s jinými klienty JMS . Pokud je typ zprávy `JMSTextMessage`, `JMSMapMessage`, `JMSStreamMessage`nebo `JMSObjectMessage`, IBM MQ se zobrazí po podrobnostech formátování zprávy. IBM MQ se zabývá rozdíly v kódových stránkách a číselným kódováním na různých platformách.

Tyto typy zpráv můžete použít k výměně zpráv s aplikacemi jiného typu nežJMS . Chcete-li to provést, musíte pochopit, jak jsou tyto zprávy vytvářeny produktem IBM MQ classes for JMS. Je možné, že budete moci upravit aplikaci, která není produktemJMS , aby se zprávy interpretoval; viz “Mapování zpráv produktu JMS na zprávy produktu IBM MQ” na stránce 126.

Výhodou použití jednoho z těchto typů zpráv je, že programování klienta JMS nezávisí na typu aplikace, se kterou si vyměňuje zprávy. Nevýhodou je, že by mohla vyžadovat úpravu jiného programu a možná nebudete moci změnit jiný program.

Alternativním přístupem je napsat klientskou aplikaci JMS , která dokáže pracovat s existujícími formáty zpráv. Často existující zprávy mají pevný formát a obsahují směsici neformátovaných dat, textu a čísel. Použijte kroky v této úloze a příklad klienta JMS v produktu “Psaní tříd pro zapouzdření rozvržení záznamu v produktu JMSBytesMessage” na stránce 167jako výchozí bod pro sestavení klienta produktu JMS , který může vyměňovat formátované záznamy s aplikacemi jiného typu nežJMS .

Postup

1. Definujte rozvržení záznamu, nebo použijte jednu z předdefinovaných tříd záhlaví produktu IBM MQ .

   Informace o zpracování předdefinovaných záhlaví produktu IBM MQ najdete v tématu Práce se záhlavími zpráv produktu IBM MQ.

   Obrázek 37 na stránce 165 je příklad rozvržení záznamu pevné délky, které lze zpracovat obslužným programem pro převod dat.

2. Vytvořte uživatelskou proceduru pro převod dat.

   Postupujte podle pokynů v části Psaní výstupního programu pro převod dat , abyste zapsali uživatelskou proceduru pro převod dat.

   Chcete-li vyzkoušet příklad v souboru “Psaní tříd pro zapouzdření rozvržení záznamu v produktu JMSBytesMessage” na stránce 167, pojmenujte ukončení převodu dat `MYRECORD`.
3. Zapište Java třídy pro zapouzdření rozvržení záznamu a odeslání a přijetí záznamu. Dva přístupy, které můžete provést, jsou:

   - Zapište třídu na čtení a zapíše `JMSBytesMessage` , která obsahuje záznam; viz “Psaní tříd pro zapouzdření rozvržení záznamu v produktu JMSBytesMessage” na stránce 167.
   - Zapište třídu rozšiřující `com.ibm.mq.header.Header` , abyste definovali datovou strukturu záznamu; viz Vytváření tříd pro nové typy záhlaví.
4. Rozhodněte, která kódovaná znaková sada se má vyměňovat ve zprávách.

   Viz téma Výběr přístupu ke konverzi zpráv: příjemce je dobrý.
5. Nakonfigurujte místo určení pro výměnu zpráv typu MQbez záhlaví JMS `MQRFH2` .

   Odesílající i přijímající místo určení musí být konfigurováno pro výměnu zpráv typu -type produktu MQ. Stejné místo určení můžete použít pro odeslání i příjem.

   Aplikace může přepsat vlastnost těla cílové zprávy:

   ```
   ((MQDestination)destination).setMessageBodyStyle(WMQConstants.WMQ_MESSAGE_BODY_MQ);
   ```

   Příklad v produktu “Psaní tříd pro zapouzdření rozvržení záznamu v produktu JMSBytesMessage” na stránce 167 potlačí vlastnost těla zprávy cíle a bude odeslána zpráva se stylem MQ.
6. Otestujte řešení pomocí aplikací produktu JMS a jiných aplikací než produktuJMS .

   Užitečné nástroje pro testování ukončení konverze dat jsou:

   - Ukázkový program `amqsgetc0.c` je užitečný k testování příjmu zprávy odeslané klientem JMS . Viz navrhované úpravy, které používají ukázkové záhlaví, `RECORD.h`, v produktu Obrázek 38 na stránce 166. S úpravami obdrží produkt `amqsgetc0.c` zprávu odeslanou příkladem klienta JMS , `TryMyRecord.java` ; viz “Psaní tříd pro zapouzdření rozvržení záznamu v produktu JMSBytesMessage” na stránce 167.
   - Ukázkový program pro procházení IBM MQ `amqsbcg0.c`je užitečný ke kontrole obsahu záhlaví zprávy, záhlaví JMS , `MQRFH2`a obsahu zprávy.
   - Program **`rfhutil`** , který byl dříve k dispozici v balíku SupportPac IH03, umožňuje zachycení a uložení testovacích zpráv v souborech a jejich použití k řízení toků zpráv. Výstupní zprávy lze také číst a zobrazovat v různých formátech. Formáty zahrnují dva typy XML, stejně jako porovnání oproti zakladači COBOL. Data mohou být ve formátu EBCDIC nebo ASCII. Do zprávy lze přidat záhlaví RFH2 , než bude odeslána zpráva.

   Pokusíte-li se o příjem zpráv pomocí upraveného ukázkového programu `amqsgetc0.c` a získání chyby s kódem příčiny `2080`, zkontrolujte, zda má zpráva `MQRFH2`. Úpravy předpokládají, že zpráva byla odeslána do místa určení, které nespecifikuje žádné `MQRFH2`.

```
struct RECORD { MQCHAR StrucID[4];
                MQLONG Version;
                                MQLONG StructLength;
                                MQLONG Encoding;
                                MQLONG CodeCharSetId;
                                MQCHAR Format[8];
                                MQLONG Flags;
                                MQCHAR RecordData[32];
        };
```

Obrázek 37. RECORD.h

- Deklarovat strukturu dat produktu RECORD.h

```
struct tagRECORD {
   MQCHAR4    StrucId;
   MQLONG     Version;
   MQLONG     StrucLength;
   MQLONG     Encoding;
   MQLONG     CCSID;
   MQCHAR8    Format;
   MQLONG     Flags;
   MQCHAR32   RecordData;
 };
typedef struct tagRECORD RECORD;
   typedef RECORD MQPOINTER PRECORD;
   RECORD record;
   PRECORD pRecord = &(record);
```

- Upravte volání MQGET tak, aby používal RECORD,

  1. Před úpravou:

```
    MQGET(Hcon,            /* connection handle     */
          Hobj,            /* object handle         */
          &md,             /* message descriptor    */
          &gmo,            /* get message options   */
          buflen,          /* buffer length         */
          buffer,          /* message buffer        */
          &messlen,        /* message length        */
          &CompCode,       /* completion code       */
          &Reason);        /* reason code           */
```

  2. Po úpravě:

```
    MQGET(Hcon,            /* connection handle     */
          Hobj,            /* object handle         */
          &md,             /* message descriptor    */
          &gmo,            /* get message options   */
          sizeof(RECORD),  /* buffer length         */
          pRecord,         /* message buffer        */
          &messlen,        /* message length        */
          &CompCode,       /* completion code       */
          &Reason);        /* reason code           */
```

- Změnit tiskový příkaz,

  1. Zdroj:

```
buffer[messlen] = '\0';            /* add terminator          */
printf("message <%s>\n", buffer);
```

  2. Do:

```
/* buffer[messlen] = '\0';            add terminator          */
printf("ccsid <%d>, flags <%d>, message <%32.32s>\n \0",
        md.CodedCharSetId, record.Flags, record.RecordData);
```

Obrázek 38. Upravit amqsget0.c

Související pojmy

Metody konverze zpráv produktu JMS

Návrháři aplikací produktu JMS jsou otevřeny pro řadu přístupů k převodu dat. Tyto přístupy se nevylučují; některé aplikace budou pravděpodobně používat kombinaci těchto přístupů. Pokud si aplikace vyměňuje pouze text nebo si vyměňuje zprávy pouze s jinými aplikacemi produktu JMS , obvykle nepovažujete převod dat za normální. Převod dat se provádí automaticky pro vás, příkazem IBM MQ.

Převod a kódování zpráv klienta JMS

Jsou zde uvedeny metody, které se používají k provedení konverze a kódování zpráv klienta JMS , s příklady kódu jednotlivých typů konverze.

Převod dat správce front
Převod dat správce front byl vždy k dispozici pro aplikace typu non-JMS , které přijímají zprávy od klientů JMS . Od verze 7.0používají klienti produktu JMS , kteří přijímají zprávy, také převod dat správce front. Počínaje verzí 7.0.1.5nebo 7.0.1.4 s opravou APAR IC72897je konverze dat správce front volitelná.

Související odkazy

Typy a konverze zpráv produktu JMS
Výběr typu zprávy ovlivňuje váš přístup k převodu zpráv. Interakce zprávy o převodu zpráv a typu zprávy jsou popsány pro typy zpráv JMS , `JMSObjectMessage`, `JMSTextMessage`, `JMSMapMessage`, `JMSStreamMessage`a `JMSBytesMessage`.

Související informace

Obslužný program pro vytvoření kódu ukončení převodu

Psaní tříd pro zapouzdření rozvržení záznamu v produktu JMSBytesMessage
Účelem této úlohy je prozkoumat například způsob, jak kombinovat převod dat a pevné rozvržení záznamu v produktu `JMSBytesMessage`. V úloze vytvoříte některé třídy Java pro výměnu vzorové struktury záznamu v `JMSBytesMessage`. Můžete upravit příklad pro zápis tříd za účelem výměny jiných struktur záznamů.

`JMSBytesMessage` je nejlepší volbou typu zprávy rozhraní JMS pro výměnu záznamů datového typu smíšených dat s jinými programy nežJMS . Neobsahuje žádná další data vložená do těla zprávy poskytovatelem JMS . Je proto nejlepším volbou typu zprávy, který má být použit, pokud klientský program JMS spolupracuje s existujícím programem IBM MQ . Hlavní úkol při použití `JMSBytesMessage` je v souladu s kódováním a znakovou sadou očekávanou jiným programem. Řešením je vytvořit třídu, která bude zapouzdřit záznam. Třída, která zapouzdřuje čtení a zapisuje `JMSBytesMessage`pro určitý typ záznamu, usnadňuje odesílání a příjem záznamů v pevném formátu v programu JMS . Díky zachycení obecných aspektů rozhraní v abstraktní třídě může být velká část řešení znovu použita pro různé formáty záznamu. Ve třídách, které rozšiřují abstraktní generickou třídu, lze implementovat různé formáty záznamů.

Alternativním přístupem je rozšíření třídy `com.ibm.mq.headers.Header` . Třída `Header` má metody, jako např. `addMQLONG`, aby vytvářel formát záznamu v deklarativním způsobem. Nevýhodou použití třídy `Header` je získávání a nastavení atributů pomocí složitějšího interpretačního rozhraní. Oba přístupy vedou ke stejnému množství kódu aplikace.

`JMSBytesMessage` může zapouzdřit pouze jeden formát, kromě `MQRFH2`, v jedné zprávě, pokud každý záznam nepoužívá stejný formát, kódovanou znakovou sadu a kódování. Formát, kódování a znaková sada `JMSBytesMessage` jsou vlastnosti všech zpráv, které následují za serverem `MQRFH2`. Příklad je zapsán za předpokladu, že produkt `JMSBytesMessage` obsahuje pouze jeden záznam uživatele.

Než začnete

1. Vaše odbornost: musíte být obeznámeni s programováním Java a JMS. O nastavení vývojového prostředí produktu Java nejsou k dispozici žádné pokyny. Je výhodné, že jste napsali program pro výměnu `JMSTextMessage`, `JMSStreamMessage`nebo `JMSMapMessage`. Poté můžete zobrazit rozdíly ve výměně zpráv pomocí produktu `JMSBytesMessage`.
2. Příklad vyžaduje IBM WebSphere MQ 7.0.
3. Příklad byl vytvořen s použitím perspektivy produktu Java pracovní plochy Eclipse . Vyžaduje prostředí JRE 6.0 nebo vyšší. K vývoji a spouštění tříd produktu Java můžete použít perspektivu produktu Java v prostředí IBM MQ Explorer. Případně použijte své vlastní vývojové prostředí produktu Java .
4. Použití Průzkumníka IBM MQ provádí nastavení testovacího prostředí a ladění, je jednodušší než použití obslužných programů příkazového řádku.

Informace o této úloze

Jste vedeni vytvořením dvou tříd: RECORD a MyRecord. Společně tyto dvě třídy zapouzdřují záznam v pevném formátu. Mají metody pro získání a nastavení atributů. Metoda get čte záznam z JMSBytesMessage a metoda put zapíše záznam do JMSBytesMessage.

Účelem úlohy není vytvoření třídy jakosti výroby, kterou lze použít opakovaně. Můžete se rozhodnout použít příklady v úloze, abyste mohli začít pracovat na svých vlastních třídách. Účelem této úlohy je poskytnout vám naváděcí poznámky, především o použití znakových sad, formátů a kódování při použití JMSBytesMessage. Je popsán každý krok při vytváření tříd a jsou popsány aspekty použití JMSBytesMessage, které se někdy přehlížejí.

Třída RECORD je abstraktní a definuje některá společná pole pro záznam uživatele. Společná pole jsou modelována na standardním rozvržení záhlaví IBM MQ , které má zvýrazňovač, verzi a délku pole. Pole kódování, znaková sada a formát, která je nalezena v mnoha záhlavích IBM MQ , jsou vynechána. Další záhlaví nemůže následovat uživatelem definovaný formát. Třída MyRecord , která rozšiřuje třídu RECORD tak, že doslova rozšiřuje záznam o další pole uživatele. Soubor JMSBytesMessagevytvořený třídami může být zpracován uživatelskou procedurou pro převod dat správce front.

Produkt “Třídy použité ke spuštění příkladu” na stránce 174 obsahuje úplný výpis položek RECORD a MyRecord. Obsahuje také výpisy dalších tříd "lešení" k testování produktů RECORD a MyRecord. Další třídy jsou:

TryMyRecord
Hlavní program pro testování RECORD a MyRecord.

EndPoint
Abstraktní třída, která zapouzdřuje připojení JMS , místo určení a relaci v jedné třídě. Jeho rozhraní právě odpovídá potřebám testování tříd RECORD a MyRecord . Není zavedený vzor návrhu pro zápis aplikací JMS .

Poznámka: Třída Endpoint obsahuje tento řádek kódu po vytvoření místa určení:

```
((MQDestination)destination).setReceiveConversion
                            (WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR);
```

V 7.0, z 7.0.1.5, je nutné zapnout převod správce front. Podle výchozího nastavení je tato volba zakázána. Ve verzi 7.0je standardně povolen převod na správce front 7.0.1.4 a tento řádek kódu způsobuje chybu.

MyProducer a MyConsumer
Třídy, které rozšiřují prostor EndPointa vytvářejí MessageConsumer a MessageProducer, jsou připojeny a připraveny přijímat požadavky.

Všechny třídy dohromady tvoří úplnou aplikaci, se kterou můžete sestavit a experimentovat s cílem pochopit, jak používat převod dat v produktu JMSBytesMessage.

Postup

1. Vytvořte abstraktní třídu pro zapouzdření standardních polí v záhlaví IBM MQ s výchozím konstruktorem. Později rozšíříte třídu tak, abyste přizpůsobili záhlaví vašim požadavkům.

```
public abstract class RECORD implements Serializable {
   private static final long serialVersionUID = -1616617232750561712L;
   protected final static int UTF8 = 1208;
   protected final static int MQLONG_LENGTH = 4;
   protected final static int RECORD_STRUCT_ID_LENGTH = 4;
   protected final static int RECORD_VERSION_1 = 1;
   protected final String RECORD_STRUCT_ID = "BLNK";
   protected final String RECORD_TYPE = "BLANK   ";
   private String structID = RECORD_STRUCT_ID;
   private int version = RECORD_VERSION_1;
   private int structLength = RECORD_STRUCT_ID_LENGTH + MQLONG_LENGTH * 2;
   private int headerEncoding = WMQConstants.WMQ_ENCODING_NATIVE;
   private String headerCharset = "UTF-8";
   private String headerFormat = RECORD_TYPE;
```

```
public RECORD() {
    super();
}
```

Poznámka:

a. Atributy, `structID` až `nextFormat`, jsou vypsány v pořadí, ve kterém jsou uvedeny ve standardním záhlaví zprávy IBM MQ .

b. Atributy, `format`, `messageEncoding`a `messageCharset`popisují samotné záhlaví a nejsou součástí záhlaví.

c. Musíte se rozhodnout, zda se má uložit identifikátor kódované znakové sady nebo znaková sada záznamu. Produkt Java používá znakové sady a IBM MQ zprávy používají identifikátory kódované znakové sady. Ukázkový kód používá znakové sady.

d. `int` je serializován do `MQLONG` pomocí IBM MQ. `MQLONG` je 4 bajty.

2. Vytvoření metod getter a setter pro soukromé atributy.

a) Vytvořte nebo vygenerujte metody getter:

```
 public String getHeaderFormat() { return headerFormat; }
public int getHeaderEncoding() { return headerEncoding;    }
public String getMessageCharset() { return headerCharset; }
public int getMessageEncoding() { return headerEncoding; }
public String getStructID() { return structID; }
public int getStructLength() { return structLength; }
public int getVersion() { return version; }
```

b) Vytvořte nebo vygenerujte metody setter:

```
 public void setHeaderCharset(String charset) {
    this.headerCharset = charset; }
public void setHeaderEncoding(int encoding) {
    this.headerEncoding = encoding; }
public void setHeaderFormat(String headerFormat) {
    this.headerFormat = headerFormat; }
public void setStructID(String structID) {
    this.structID = structID; }
public void setStructLength(int structLength) {
    this.structLength = structLength; }
public void setVersion(int version) {
    this.version = version; }
}
```

3. Vytvořte konstruktor pro vytvoření instance RECORD z `JMSBytesMessage`.

```
 public RECORD(BytesMessage message) throws JMSException, IOException,
        MQDataException {
    super();
    setHeaderCharset(message.getStringProperty(WMQConstants.JMS_IBM_CHARACTER_SET));
    setHeaderEncoding(message.getIntProperty(WMQConstants.JMS_IBM_ENCODING));
    byte[] structID = new byte[RECORD_STRUCT_ID_LENGTH];
    message.readBytes(structID, RECORD_STRUCT_ID_LENGTH);
    setStructID(new String(structID, getMessageCharset()));
    setVersion(message.readInt());
    setStructLength(message.readInt());
}
```

Poznámka:

a. Hodnoty `messageCharset` a `messageEncoding`jsou zachyceny ve vlastnostech zprávy, protože přepisují hodnoty nastavené pro místo určení. `format` není aktualizován. Příklad nekontroluje chybu. Je-li volán konstruktor `Record(BytesMessage)` , předpokládá se, že `JMSBytesMessage` je typ zprávy RECORD . Řádek "`setStructID(new String(structID, getMessageCharset()))`" nastaví zvýrazňovač očí.

b. Čáry kódu, které dokončují deserializační pole metody ve zprávě, v pořadí aktualizující výchozí hodnoty nastavené v instanci RECORD.

4. Vytvořte metodu put pro zápis polí záhlaví do JMSBytesMessage.

```
protected BytesMessage put(MyProducer myProducer) throws IOException,
       JMSException, UnsupportedEncodingException {
   setHeaderEncoding(myProducer.getEncoding());
   setHeaderCharset(myProducer.getCharset());
   myProducer.setMQClient(true);
   BytesMessage bytes = myProducer.session.createBytesMessage();
   bytes.setStringProperty(WMQConstants.JMS_IBM_FORMAT, getHeaderFormat());
   bytes.setIntProperty(WMQConstants.JMS_IBM_ENCODING, getHeaderEncoding());
   bytes.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET,
                        myProducer.getCCSID());
   bytes.writeBytes(String.format("%1$-" + RECORD_STRUCT_ID_LENGTH + "."
                   + RECORD_STRUCT_ID_LENGTH + "s", getStructID())
                   .getBytes(getMessageCharset()), 0, RECORD_STRUCT_ID_LENGTH);
   bytes.writeInt(getVersion());
   bytes.writeInt(getStructLength());
   return bytes;
}
```

Poznámka:

a. MyProducer zapouzdřuje JMS Connection, Destination, Sessiona MessageProducer v jediné třídě. MyConsumer, použije se později, zapouzdří JMS Connection, Destination, Sessiona MessageConsumer v jedné třídě.

b. Pokud je pro systém JMSBytesMessagekódování jiné než Native, musí být kódování nastaveno ve zprávě. Kódování cíle se zkopíruje do atributu kódování zpráv JMS_IBM_CHARACTER_SETa uloží se jako atribut třídy RECORD .

i) "setMessageEncoding(myProducer.getEncoding());" volá příkaz "(((MQDestination) destination).getIntProperty(WMQConstants.WMQ_ENCODING));" k získání cílového kódování.

ii) Produkt "Bytes.setIntProperty(WMQConstants.JMS_IBM_ENCODING, getMessageEncoding());" nastavuje kódování zpráv.

c. Znaková sada použitá k transformaci textu na bajty se získá z cíle a uloží se jako atribut třídy RECORD . Není nastavena ve zprávě, protože ji nepoužívá IBM MQ classes for JMS při zápisu JMSBytesMessage.

Volání "messageCharset = myProducer.getCharset();"

```
public String getCharset() throws UnsupportedEncodingException,
        JMSException {
    return CCSID.getCodepage(getCCSID());
}
```

Z identifikátoru kódované znakové sady získá znakovou sadu Java .

" CCSID.getCodepage(ccsid) " je v balíku com.ibm.mq.headers. Produkt ccsid se získává z jiné metody v produktu MyProducer, která se dotazuje na místo určení:

```
public int getCCSID() throws JMSException {
    return (((MQDestination) destination)
            .getIntProperty(WMQConstants.WMQ_CCSID));
}
```

d. Volba "myProducer.setMQClient(true);" přepíše cílové nastavení pro typ klienta a vynutí jej na IBM MQ MQI client. Možná dáte přednost vynechání této řádky kódu, protože zamlžuje administrativní chybu konfigurace.

Volání "myProducer.setMQClient(true);" :

```
((MQDestination) destination).setTargetClient(WMQConstants.WMQ_TARGET_DEST_MQ); }
if (!getMQDest()) setMQBody();
```

Tento kód má vedlejší účinek nastavení stylu textu IBM MQ na nespecifikovaný, pokud musí přepsat nastavení JMS.

Poznámka:

IBM MQ classes for JMS zapisuje formát, kódování a identifikátor znakové sady zprávy do deskriptoru zpráv, MQMDnebo do záhlaví JMS , MQRFH2. Závisí na tom, zda má zpráva tělo stylu IBM MQ . Nenastavujte pole MQMD ručně.

Existuje metoda pro ruční nastavení vlastností deskriptoru zpráv. Používá vlastnosti produktu JMS_IBM_MQMD_* . Chcete-li nastavit vlastnosti produktu JMS_IBM_MQMD_* , musíte nastavit vlastnost cíle WMQ_MQMD_WRITE_ENABLED :

```
((MQDestination)destination).setMQMDWriteEnabled(true);
```

Chcete-li si přečíst vlastnosti, musíte nastavit cílovou vlastnost WMQ_MQMD_READ_ENABLED.

JMS_IBM_MQMD_* použijte pouze v případě, že plně ovládíte celý informační obsah zprávy. Na rozdíl od vlastností produktu JMS_IBM_* neřídí vlastnosti JMS_IBM_MQMD_* , jak IBM MQ classes for JMS konstruují zprávu JMS . Je možné vytvořit vlastnosti deskriptoru zpráv, které jsou v konfliktu s vlastnostmi zprávy produktu JMS .

e. Čáry kódu, které dokončí metodu, budou serializovat atributy ve třídě jako pole ve zprávě.

Atributy řetězce jsou doplněny mezerami. Řetězce jsou převedeny na bajty pomocí znakové sady definované pro záznam a oříznuty na délku polí zprávy.

5. Dokončete třídu přidáním importů.

```
 package com.ibm.mq.id;
import java.io.IOException;
import java.io.Serializable;
import java.io.UnsupportedEncodingException;
import javax.jms.BytesMessage;
import javax.jms.JMSException;
import com.ibm.mq.constants.MQConstants;
import com.ibm.mq.headers.MQDataException;
import com.ibm.msg.client.wmq.WMQConstants;
```

6. Vytvořte třídu, abyste rozšířily třídu RECORD tak, aby zahrnovala další pole. Zahrnout výchozí konstruktor.

```
public class MyRecord extends RECORD {
   private static final long serialVersionUID = -370551723162299429L;
   private final static int FLAGS = 1;
   private final static String STRUCT_ID = "MYRD";
   private final static int DATA_LENGTH = 32;
   private final static String FORMAT = "MYRECORD";
   private int flags = FLAGS;
   private String recordData = "ABCDEFGHIJKLMNOPQRSTUVWXYZ012345";

   public MyRecord() {
       super();
       super.setStructID(STRUCT_ID);
       super.setHeaderFormat(FORMAT);
       super.setStructLength(super.getStructLength() + MQLONG_LENGTH
               + DATA_LENGTH);
   }
```

Poznámka:

a. Podtřída RECORD , MyRecord, upravuje velikost záhlaví, formát a délku záhlaví.

7. Vytvořte nebo vygenerujte metody getter a setter.
   a) Vytvořte metody getter:

```
    public int getFlags() { return flags; }
   public String getRecordData() { return recordData; } .
```

   b) Vytvořte metody Setter:

```
    public void setFlags(int flags) {
       this.flags = flags; }
   public void setRecordData(String recordData) {
       this.recordData = recordData; }
}
```

8. Vytvořte konstruktor pro vytvoření instance `MyRecord` z `JMSBytesMessage`.

```
 public MyRecord(BytesMessage message) throws JMSException, IOException,
        MQDataException {
    super(message);
    setFlags(message.readInt());
    byte[] recordData = new byte[DATA_LENGTH];
    message.readBytes(recordData, DATA_LENGTH);
    setRecordData(new String(recordData, super.getMessageCharset()));
}
```

   Poznámka:

   a. Pole, která tvoří standardní šablonu zprávy, jsou přečteny nejprve třídou `RECORD` .

   b. Text `recordData` se převede na `String` s použitím vlastnosti znakové sady pro zprávu.

9. Vytvořte statickou metodu pro získání zprávy od spotřebitele a vytvořte novou instanci produktu `MyRecord` .

```
 public static MyRecord get(MyConsumer myConsumer) throws JMSException,
        MQDataException, IOException {
    BytesMessage message = (BytesMessage) myConsumer.receive();
    return new MyRecord(message);
}
```

   Poznámka:

   a. V příkladu z důvodu stručnosti je konstruktor `MyRecord(BytesMessage)` volán ze statické metody get. Obvykle se můžete oddělit od přijetí zprávy od vytvoření nové instance `MyRecord` .

10. Vytvořte metodu put k připojení polí zákazníků k `JMSBytesMessage` obsahujícímu záhlaví zprávy.

```
 public BytesMessage put(MyProducer myProducer) throws JMSException,
        IOException {
    BytesMessage bytes = super.put(myProducer);
    bytes.writeInt(getFlags());
    bytes.writeBytes(String.format("%1$-" + DATA_LENGTH + "."
                    + DATA_LENGTH + "s",getRecordData())
                    .getBytes(super.getMessageCharset()), 0, DATA_LENGTH);
    myProducer.send(bytes);
    return bytes;
}
```

   Poznámka:

   a. Metoda volá v kódu serializovat atributy ve třídě MyRecord jako pole ve zprávě.

   - Atribut `recordData` `String` je doplněn mezerami, převedený na bajty pomocí znakové sady definované pro záznam a zkrácen na délku pole `RecordData` .

11. Dokončete třídu přidáním příkazů include.

```
 package com.ibm.mq.id;
import java.io.IOException;
import javax.jms.BytesMessage;
import javax.jms.JMSException;
import com.ibm.mq.headers.MQDataException;
```

Výsledky

Výsledky:

- Výsledky ze spuštění třídy `TryMyRecord` :
  - Odesílá se zpráva v kódované znakové sadě 37 a při použití uživatelské procedury pro převod správce front:

```
Out flags 1 text ABCDEFGHIJKLMNOPQRSTUVWXYZ012345 Encoding 546 CCSID 37 MQ true
Out flags 1 text ABCDEFGHIJKLMNOPQRSTUVWXYZ012345 Encoding 546 CCSID 37 MQ true
In flags 1 text ABCDEFGHIJKLMNOPQRSTUVWXYZ012345 Encoding 273 CCSID UTF-8
```

  - Odesílá se zpráva v kódované znakové sadě 37 a nepoužívající uživatelskou proceduru pro převod správce front:

```
Out flags 1 text ABCDEFGHIJKLMNOPQRSTUVWXYZ012345 Encoding 546 CCSID 37 MQ true
Out flags 1 text ABCDEFGHIJKLMNOPQRSTUVWXYZ012345 Encoding 546 CCSID 37 MQ true
In flags 1 text ABCDEFGHIJKLMNOPQRSTUVWXYZ012345 Encoding 546 CCSID IBM037
```

- Výsledkem úpravy třídy `TryMyRecord` není přijetí zprávy a místo toho přijímá ji s použitím upraveného vzorku `amqsget0.c` . Upravená ukázka přijímá formátovaný záznam; viz Obrázek 38 na stránce 166 v “Výměna formátovaného záznamu s aplikací jinou nežJMS” na stránce 163.
  - Odesílá se zpráva v kódované znakové sadě 37 a při použití uživatelské procedury pro převod správce front:

```
Sample AMQSGET0 start
ccsid <850>, flags <1>, message <ABCDEFGHIJKLMNOPQRSTUVWXYZ012345>
no more messages
Sample AMQSGET0 end
```

  - Odesílá se zpráva v kódované znakové sadě 37 a nepoužívající uživatelskou proceduru pro převod správce front:

```
Sample AMQSGET0 start
MQGET ended with reason code 2110
ccsid <37>, flags <1>, message <--+-+ãÃ++ÐÊËÈiÐÎÐ+ÔÒõõµþÞÚ-±=¾¶§>
no more messages
Sample AMQSGET0 end
```

Chcete-li vyzkoušet příklad a experimentovat s různými kódovými stránkami a s uživatelskou procedurou pro převod dat. Vytvořte třídy Java , nakonfigurujte IBM MQa spusťte hlavní program `TryMyRecord` ; viz Obrázek 39 na stránce 174.

1. Chcete-li spustit tento příklad, nakonfigurujte prostor IBM MQ a JMS . Pokyny jsou určeny ke spuštění příkladu na serveru Windows.
   a. Vytvoření správce front

```
crtmqm -sa -u SYSTEM.DEAD.LETTER.QUEUE QM1
strmqm QM1
```

   b. Vytvoření fronty

```
echo DEFINE QL('Q1') REPLACE | runmqsc QM1
```

c. Vytvořit adresář JNDI

```
cd c:\
md JNDI-Directory
```

d. Přejděte do adresáře bin produktu JMS .

Musí být spuštěn administrativní program produktu JMS . Cesta je *MQ_INSTALLATION_PATH*\java\bin.

e. Vytvořte následující definice JMS v souboru s názvem JMSQM1Q1.txt

```
DEF CF(QM1) PROVIDERVERSION(7) QMANAGER(QM1)
DEF Q(Q1) CCSID(37) ENCODING(RRR) MSGBODY(MQ) QMANAGER(QM1) QUEUE(Q1) TARGCLIENT(MQ)
VERSION(7)
END
```

f. Spusťte program JMSAdmin a vytvořte prostředky produktu JMS .

```
JMSAdmin < JMSQM1Q1.txt
```

2. Definice, které jste vytvořili pomocí Průzkumníka IBM MQ , můžete vytvářet, měnit a procházet.
3. Spusťte příkaz TryMyRecord.

Třídy použité ke spuštění příkladu

Třídy uvedené v obrázcích Obrázek 39 na stránce 174 až Obrázek 44 na stránce 178 jsou k dispozici také v komprimovaném souboru; stáhněte soubor jm25529_.zip nebo jm25529_.tar.gz.

```
package com.ibm.mq.id;
public class TryMyRecord {
    public static void main(String[] args) throws Exception {
        MyProducer producer = new MyProducer();
        MyRecord outrec = new MyRecord();
        System.out.println("Out flags " + outrec.getFlags() + " text "
                + outrec.getRecordData() + " Encoding "
                + producer.getEncoding() + " CCSID " + producer.getCCSID()
                + " MQ " + producer.getMQDest());
        outrec.put(producer);
        System.out.println("Out flags " + outrec.getFlags() + " text "
                + outrec.getRecordData() + " Encoding "
                + producer.getEncoding() + " CCSID " + producer.getCCSID()
                + " MQ " + producer.getMQDest());
        MyRecord inrec = MyRecord.get(new MyConsumer());
        System.out.println("In  flags " + inrec.getFlags() + " text "
                + inrec.getRecordData() + " Encoding "
                + inrec.getMessageEncoding() + " CCSID "
                + inrec.getMessageCharset());
    }
}
```

Obrázek 39. TryMyRecord

```
package com.ibm.mq.id;
import java.io.IOException;
import java.io.Serializable;
import java.io.UnsupportedEncodingException;
import javax.jms.BytesMessage;
import javax.jms.JMSException;
import com.ibm.mq.constants.MQConstants;
import com.ibm.mq.headers.MQDataException;
import com.ibm.msg.client.wmq.WMQConstants;

public abstract class RECORD implements Serializable {
    private static final long serialVersionUID = -1616617232750561712L;
    protected final static int UTF8 = 1208;
    protected final static int MQLONG_LENGTH = 4;
    protected final static int RECORD_STRUCT_ID_LENGTH = 4;
    protected final static int RECORD_VERSION_1 = 1;
    protected final String RECORD_STRUCT_ID = "BLNK";
    protected final String RECORD_TYPE = "BLANK   ";
    private String structID = RECORD_STRUCT_ID;
    private int version = RECORD_VERSION_1;
    private int structLength = RECORD_STRUCT_ID_LENGTH + MQLONG_LENGTH * 2;
    private int headerEncoding = WMQConstants.WMQ_ENCODING_NATIVE;
    private String headerCharset = "UTF-8";
    private String headerFormat = RECORD_TYPE;

    public RECORD() {
        super();
    }

    public RECORD(BytesMessage message) throws JMSException, IOException,
            MQDataException {
        super();
        setHeaderCharset(message.getStringProperty(WMQConstants.JMS_IBM_CHARACTER_SET));
        setHeaderEncoding(message.getIntProperty(WMQConstants.JMS_IBM_ENCODING));
        byte[] structID = new byte[RECORD_STRUCT_ID_LENGTH];
        message.readBytes(structID, RECORD_STRUCT_ID_LENGTH);
        setStructID(new String(structID, getMessageCharset()));
        setVersion(message.readInt());
        setStructLength(message.readInt());
    }

    public String getHeaderFormat() { return headerFormat; }
    public int getHeaderEncoding() { return headerEncoding;    }
    public String getMessageCharset() { return headerCharset; }
    public int getMessageEncoding() { return headerEncoding; }
    public String getStructID() { return structID; }
    public int getStructLength() { return structLength; }
    public int getVersion() { return version; }


    protected BytesMessage put(MyProducer myProducer) throws IOException,
            JMSException, UnsupportedEncodingException {
        setHeaderEncoding(myProducer.getEncoding());
        setHeaderCharset(myProducer.getCharset());
        myProducer.setMQClient(true);
        BytesMessage bytes = myProducer.session.createBytesMessage();
        bytes.setStringProperty(WMQConstants.JMS_IBM_FORMAT, getHeaderFormat());
        bytes.setIntProperty(WMQConstants.JMS_IBM_ENCODING, getHeaderEncoding());
        bytes.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET,
                             myProducer.getCCSID());
        bytes.writeBytes(String.format("%1$-" + RECORD_STRUCT_ID_LENGTH + "."
                         + RECORD_STRUCT_ID_LENGTH + "s", getStructID())
                         .getBytes(getMessageCharset()), 0, RECORD_STRUCT_ID_LENGTH);
        bytes.writeInt(getVersion());
        bytes.writeInt(getStructLength());
        return bytes;
    }

    public void setHeaderCharset(String charset) {
        this.headerCharset = charset; }
    public void setHeaderEncoding(int encoding) {
        this.headerEncoding = encoding; }
    public void setHeaderFormat(String headerFormat) {
        this.headerFormat = headerFormat; }
    public void setStructID(String structID) {
        this.structID = structID; }
    public void setStructLength(int structLength) {
        this.structLength = structLength; }
    public void setVersion(int version) {
        this.version = version; }
}
```

Obrázek 40. RECORD

```
package com.ibm.mq.id;
import java.io.IOException;
import javax.jms.BytesMessage;
import javax.jms.JMSException;
import com.ibm.mq.headers.MQDataException;

public class MyRecord extends RECORD {
    private static final long serialVersionUID = -370551723162299429L;
    private final static int FLAGS = 1;
    private final static String STRUCT_ID = "MYRD";
    private final static int DATA_LENGTH = 32;
    private final static String FORMAT = "MYRECORD";
    private int flags = FLAGS;
    private String recordData = "ABCDEFGHIJKLMNOPQRSTUVWXYZ012345";

    public MyRecord() {
        super();
        super.setStructID(STRUCT_ID);
        super.setHeaderFormat(FORMAT);
        super.setStructLength(super.getStructLength() + MQLONG_LENGTH
                + DATA_LENGTH);
    }

    public MyRecord(BytesMessage message) throws JMSException, IOException,
            MQDataException {
        super(message);
        setFlags(message.readInt());
        byte[] recordData = new byte[DATA_LENGTH];
        message.readBytes(recordData, DATA_LENGTH);
        setRecordData(new String(recordData, super.getMessageCharset()));
    }

    public static MyRecord get(MyConsumer myConsumer) throws JMSException,
            MQDataException, IOException {
        BytesMessage message = (BytesMessage) myConsumer.receive();
        return new MyRecord(message);
    }

    public int getFlags() { return flags; }
    public String getRecordData() { return recordData; } .


    public BytesMessage put(MyProducer myProducer) throws JMSException,
            IOException {
        BytesMessage bytes = super.put(myProducer);
        bytes.writeInt(getFlags());
        bytes.writeBytes(String.format("%1$-" + DATA_LENGTH + "."
                          + DATA_LENGTH + "s",getRecordData())
                          .getBytes(super.getMessageCharset()), 0, DATA_LENGTH);
        myProducer.send(bytes);
        return bytes;
    }

    public void setFlags(int flags) {
        this.flags = flags; }
    public void setRecordData(String recordData) {
        this.recordData = recordData; }
}
```

Obrázek 41. MyRecord

```
package com.ibm.mq.id;
import java.io.UnsupportedEncodingException;
import javax.jms.Connection;
import javax.jms.ConnectionFactory;
import javax.jms.Destination;
import javax.jms.JMSException;
import javax.jms.Session;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import com.ibm.mq.headers.CCSID;
import com.ibm.mq.jms.MQDestination;
import com.ibm.msg.client.wmq.WMQConstants;
public abstract class EndPoint {
    public Context ctx;
    public ConnectionFactory cf;
    public Connection connection;
    public Destination destination;
    public Session session;
    protected EndPoint() throws NamingException, JMSException {
        System.setProperty("java.naming.provider.url", "file:/C:/JNDI-Directory");
        System.setProperty("java.naming.factory.initial",
                           "com.sun.jndi.fscontext.RefFSContextFactory");
        ctx = new InitialContext();
        cf = (ConnectionFactory) ctx.lookup("QM1");
        connection = cf.createConnection();
        destination = (Destination) ctx.lookup("Q1");
        ((MQDestination)destination).setReceiveConversion
                                    (WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR);
        session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE); }
    protected EndPoint(String cFactory, String dest) throws NamingException,
            JMSException {
        System.setProperty("java.naming.provider.url", "file:/C:/JNDI-Directory");
        System.setProperty("java.naming.factory.initial",
                           "com.sun.jndi.fscontext.RefFSContextFactory");
        ctx = new InitialContext();
        cf = (ConnectionFactory) ctx.lookup(cFactory);
        connection = cf.createConnection();
        destination = (Destination) ctx.lookup(dest);
        ((MQDestination)destination).setReceiveConversion
                                    (WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR);
        session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);  }
    public int getCCSID() throws JMSException {
        return (((MQDestination) destination)
                .getIntProperty(WMQConstants.WMQ_CCSID));  }
    public String getCharset() throws UnsupportedEncodingException,
             JMSException {
        return CCSID.getCodepage(getCCSID());  }
    public int getEncoding() throws JMSException {
        return (((MQDestination) destination)
                .getIntProperty(WMQConstants.WMQ_ENCODING));  }
    public boolean getMQDest() throws JMSException {
        if ((((MQDestination) destination).getMessageBodyStyle()
              == WMQConstants.WMQ_MESSAGE_BODY_MQ)
            || ((((MQDestination) destination).getMessageBodyStyle()
                  == WMQConstants.WMQ_MESSAGE_BODY_UNSPECIFIED)
               && (((MQDestination) destination).getTargetClient()
                  == WMQConstants.WMQ_TARGET_DEST_MQ)))
            return true;
        else
            return false; }
    public void setCCSID(int ccsid) throws JMSException {
        ((MQDestination) destination).setIntProperty(WMQConstants.WMQ_CCSID,
                ccsid); }
    public void setEncoding(int encoding) throws JMSException {
        ((MQDestination) destination).setIntProperty(WMQConstants.WMQ_ENCODING,
                encoding); }
    public void setMQBody() throws JMSException {
        ((MQDestination) destination)
                    .setMessageBodyStyle(WMQConstants.WMQ_MESSAGE_BODY_UNSPECIFIED);  }
    public void setMQBody(boolean mqbody) throws JMSException {
        if (mqbody)  ((MQDestination) destination)
                    .setMessageBodyStyle(WMQConstants.WMQ_MESSAGE_BODY_MQ);
        else         ((MQDestination) destination)
                    .setMessageBodyStyle(WMQConstants.WMQ_MESSAGE_BODY_JMS);  }
    public void setMQClient(boolean mqclient) throws JMSException {
       if (mqclient){
            ((MQDestination) destination).setTargetClient(WMQConstants.WMQ_TARGET_DEST_MQ);
            if (!getMQDest()) setMQBody();
        }
       else
            ((MQDestination) destination).setTargetClient(WMQConstants.WMQ_TARGET_DEST_JMS); }
    }
```

Obrázek 42. EndPoint

```
package com.ibm.mq.id;
import javax.jms.JMSException;
import javax.jms.Message;
import javax.jms.MessageProducer;
import javax.naming.NamingException;
public class MyProducer extends EndPoint {
    public MessageProducer producer;
    public MyProducer() throws NamingException, JMSException {
        super();
        producer = session.createProducer(destination); }
    public MyProducer(String cFactory, String dest) throws NamingException,
            JMSException {
        super(cFactory, dest);
        producer = session.createProducer(destination); }
    public void send(Message message) throws JMSException {
        producer.send(message); }
}
```

Obrázek 43. MyProducer

```
package com.ibm.mq.id;
import javax.jms.JMSException;
import javax.jms.Message;
import javax.jms.MessageConsumer;
import javax.naming.NamingException;
public class MyConsumer extends EndPoint {
    public MessageConsumer consumer;
    public MyConsumer() throws NamingException, JMSException {
        super();
        consumer = session.createConsumer(destination);
        connection.start();  }
    public MyConsumer(String cFactory, String dest) throws NamingException,
            JMSException {
        super(cFactory, dest);
        consumer = session.createConsumer(destination);
        connection.start();  }
    public Message receive() throws JMSException {
        return consumer.receive();  }
}
```

Obrázek 44. MyConsumer

Vytvoření a konfigurace továren připojení a cílů v aplikaci IBM MQ classes for JMS

Aplikace IBM MQ classes for JMS může vytvářet továrny připojení a cíle tím, že je načte jako spravované objekty z oboru názvů rozhraní JNDI (Java Naming and Directory Interface) pomocí rozšíření produktu IBM JMS nebo použitím rozšíření produktu IBM MQ JMS . Aplikace může také použít rozšíření IBM JMS nebo rozšíření IBM MQ JMS k nastavení vlastností továren připojení a míst určení.

Továrny připojení a cíle jsou počáteční body v toku logiky aplikace produktu JMS . Aplikace používá objekt ConnectionFactory k vytvoření připojení k serveru systému zpráv a používá objekt Fronta nebo Téma jako cíl pro odesílání zpráv do nebo ze zdroje, ze kterého mají být přijímány zprávy. Aplikace proto musí vytvořit alespoň jednu továrnu připojení a jedno nebo více míst určení. Po vytvoření faktorie připojení nebo místa určení pak může aplikace nakonfigurovat objekt tak, že nastaví jednu nebo více jejích vlastností.

Stručně řečeno, aplikace může vytvářet a konfigurovat továrny připojení a cíle následujícími způsoby:

Použití JNDI k načtení spravovaných objektů
Administrátor může použít nástroj pro administraci produktu IBM MQ JMS , jak je popsáno v části Konfigurace objektů pomocí nástroje pro administraci produktu JMS, nebo IBM MQ Explorer , jak je popsáno v tématu Konfigurace objektů produktu JMS pomocí produktu IBM MQ Explorer, k vytváření a konfiguraci továren připojení a míst určení jako administrovaných objektů v oboru názvů JNDI. Aplikace potom může načíst spravované objekty z oboru názvů JNDI. Po načtení spravovaného objektu může aplikace v případě potřeby nastavit nebo změnit jednu či více jejích vlastností buď pomocí přípon IBM JMS , nebo rozšíření IBM MQ JMS .

Použití rozšíření produktu IBM JMS
Aplikace může používat rozšíření produktu IBM JMS k dynamickému vytváření továren připojení a cílů v běhovém prostředí. Aplikace nejprve vytvoří objekt továrny JmsFactorya poté použije metody tohoto objektu k vytvoření továren připojení a cílů. Po vytvoření faktorie připojení nebo místa určení může aplikace využívat metody zděděné z rozhraní kontextu JmsPropertyk nastavení vlastností.

Alternativně může aplikace použít identifikátor URI (Uniform Resource Identifier) k určení jedné nebo více vlastností cíle při vytváření místa určení.

Použití rozšíření produktu IBM MQ JMS

Aplikace může také použít rozšíření produktu IBM MQ JMS k dynamickému vytváření továren připojení a cílů v běhovém prostředí. Aplikace používá dodané konstruktory k vytvoření továren připojení a cílů. Po vytvoření faktorie připojení nebo místa určení může aplikace použít metody objektu k nastavení vlastností objektu. Aplikace může alternativně použít identifikátor URI k určení jedné nebo více vlastností cíle při vytváření místa určení.

Související informace

Konfigurace prostředků produktu JMS

Použití JNDI k načtení spravovaných objektů v aplikaci JMS

Chcete-li načíst spravované objekty z oboru názvů rozhraní JNDI (Java Naming and Directory Interface), aplikace JMS musí vytvořit počáteční kontext a pak pomocí metody lookup () načíst objekty.

Než bude moci aplikace načítat spravované objekty z oboru názvů JNDI, musí nejprve vytvořit spravované objekty administrátor. Administrátor může použít nástroj pro administraci produktu IBM MQ JMS nebo produkt IBM MQ Explorer k vytvoření a údržbě spravovaných objektů v oboru názvů JNDI. Další informace viz téma Konfigurace továren připojení a míst určení v oboru názvů JNDI.

Aplikační server, obvykle poskytuje své vlastní úložiště pro spravované objekty a vlastní nástroje pro vytváření a údržbu objektů.

Chcete-li načíst spravované objekty z oboru názvů JNDI, aplikace musí nejprve vytvořit počáteční kontext, jak je uvedeno v následujícím příkladu:

```
import javax.jms.*;
import javax.naming.*;
import javax.naming.directory.*;
.
.
.
String url = "ldap://server.company.com/o=company_us,c=us";
String icf = "com.sun.jndi.ldap.LdapCtxFactory";
.
java.util.Hashtable environment = new java.util.Hashtable();
environment.put(Context.PROVIDER_URL, url);
environment.put(Context.INITIAL_CONTEXT_FACTORY, icf);
Context ctx = new InitialDirContext(environment);
```

V tomto kódu mají řetězcové proměnné `url` a `icf` následující význam:

url

Uniform resource locator (URL) adresářové služby. Adresa URL může mít jeden z následujících formátů:

- `ldap://`*`hostname`*`/`*`contextName`* , pro adresářovou službu založenou na serveru LDAP
- `file:/`*`directoryPath`* , pro adresářovou službu založenou na lokálním systému souborů

if

Název třídy počáteční továrny kontextu, který může mít jednu z následujících hodnot:

- `com.sun.jndi.ldap.LdapCtxFactory`, pro adresářovou službu založenou na serveru LDAP
- `com.sun.jndi.fscontext.RefFSContextFactory`, pro adresářovou službu založenou na lokálním systému souborů

Všimněte si, že některé kombinace balíku JNDI a poskytovatele služeb LDAP (Lightweight Directory Access Protocol) mohou způsobit výskyt chyby LDAP 84. Chcete-li tento problém vyřešit, vložte před volání do kontextu InitialDirContext () následující řádek kódu:

```
environment.put(Context.REFERRAL, "throw");
```

Po získání počátečního kontextu může aplikace načíst administrované objekty z oboru názvů JNDI pomocí metody lookup (), jak je uvedeno v následujícím příkladu:

```
ConnectionFactory factory;
Queue queue;
Topic topic;
.
.
.
factory = (ConnectionFactory)ctx.lookup("cn=myCF");
queue = (Queue)ctx.lookup("cn=myQ");
topic = (Topic)ctx.lookup("cn=myT");
```

Tento kód načítá následující objekty z oboru názvů založené na protokolu LDAP:

- Objekt ConnectionFactory je svázán s názvem myCF .
- Objekt fronty svázaný s názvem myQ .
- Objekt tématu vázaný s názvem myT

Další informace o používání rozhraní JNDI najdete v dokumentaci k rozhraní JNDI poskytované společností Oracle Corporation.

Související informace

Konfigurace objektů produktu JMS pomocí produktu IBM MQ Explorer

Konfigurace objektů produktu JMS pomocí nástroje pro administraci

Konfigurace prostředků produktu JMS v produktu WebSphere Application Server

Použití rozšíření produktu IBM JMS

IBM MQ classes for JMS obsahuje sadu rozšíření rozhraní API JMS s názvem rozšíření IBM JMS . Aplikace může tato rozšíření použít k dynamickému vytváření továren připojení a cílů v běhovém prostředí a k nastavení vlastností objektů produktu IBM MQ classes for JMS . Rozšíření lze použít s libovolným poskytovatelem systému zpráv.

Rozšíření IBM JMS jsou sadou rozhraní a tříd v následujících balících:

- com.ibm.msg.client.jms
- com.ibm.msg.client.services

Balíky lze nalézt v `com.ibm.mqjms.jar` , který se nachází v *MQ_INSTALLATION_PATH*`/java/lib`.

Tato rozšíření poskytují následující funkce:

- Mechanizmus dynamického vytváření továren a cílů připojení za běhu v továrně namísto jejich načtení jako spravovaných objektů z oboru názvů rozhraní JNDI (Java Naming and Directory Interface)
- Sada metod pro nastavení vlastností objektů produktu IBM MQ classes for JMS
- Sada tříd výjimek s metodami pro získání podrobných informací o problému
- Sada metod pro řízení trasování
- Sada metod pro získání informací o verzi o produktu IBM MQ classes for JMS

Pokud se týká vytváření továren připojení a míst určení dynamicky za běhu a nastavení a získání jejich vlastností, rozšíření produktu IBM JMS poskytují alternativní sadu rozhraní k rozšíření produktu IBM MQ JMS . Vzhledem k tomu, že rozšíření produktu IBM MQ JMS jsou specifická pro poskytovatele systému zpráv produktu IBM MQ , rozšíření produktu IBM JMS nejsou specifická pro produkt IBM MQ a lze ji použít s libovolným poskytovatelem systému zpráv v rámci vrstvené architektury popsané v tématu Architektura produktuIBM MQ pro architekturu JMS.

Rozhraní com.ibm.msg.client.wmq.WMQConstants obsahuje definice konstant, které může aplikace použít při nastavování vlastností objektů produktu IBM MQ classes for JMS pomocí rozšíření produktu IBM JMS . Rozhraní obsahuje konstanty pro poskytovatele systému zpráv produktu IBM MQ a konstanty JMS , které jsou nezávislé na všech poskytovatelích systému zpráv.

Příklady kódu, které následují za předpokladu, že byly spuštěny následující příkazy importu:

```
import com.ibm.msg.client.jms.*;
import com.ibm.msg.client.services.*;
import com.ibm.msg.client.wmq.WMQConstants;
```

Vytvoření továren připojení a míst určení

Než bude aplikace moci vytvořit továrny připojení a cíle pomocí rozšíření produktu IBM JMS , musí nejprve vytvořit objekt továrny JmsFactory. Chcete-li vytvořit objekt továrny JmsFactory, aplikace volá metodu getInstance() třídy továrny JmsFactory, jak je uvedeno v následujícím příkladu:

```
JmsFactoryFactory ff = JmsFactoryFactory.getInstance(JmsConstants.WMQ_PROVIDER);
```

Parametr ve volání metody getInstance() je konstanta, která identifikuje poskytovatele systému zpráv produktu IBM MQ jako vybraného poskytovatele systému zpráv. Aplikace pak může pomocí objektu továrny JmsFactoryvytvářet továrny připojení a místa určení.

Chcete-li vytvořit továrnu na připojení, aplikace volá metodu createConnectionFactory () objektu továrny JmsFactory, jak je uvedeno v následujícím příkladu:

```
JmsConnectionFactory factory = ff.createConnectionFactory();
```

Tento příkaz vytvoří objekt továrny JmsConnections výchozími hodnotami pro všechny jeho vlastnosti, což znamená, že se aplikace připojuje k výchozímu správci front v režimu vazeb. Chcete-li, aby se aplikace připojovala v režimu klienta nebo se připojila k jinému správci front, než je výchozí správce front, musí aplikace před vytvořením připojení nastavit příslušné vlastnosti objektu továrny JmsConnection. Další informace o tom, jak to provést, viz “Nastavení vlastností objektů produktu IBM MQ classes for JMS” na stránce 182.

Třída továrny JmsFactorytaké obsahuje metody pro vytvoření továren na připojení následujících typů:

- JmsQueueConnectionFactory
- JmsTopicConnectionFactory
- Továrna JmsXAConnection
- JmsXAQueueConnectionFactory
- JmsXATopicConnectionFactory

Chcete-li vytvořit objekt fronty, volá aplikace metodu createQueue() objektu továrny JmsFactory, jak je uvedeno v následujícím příkladu:

```
JmsQueue q1 = ff.createQueue("Q1");
```

Tento příkaz vytvoří objekt JmsQueue s výchozími hodnotami pro všechny jeho vlastnosti. Objekt představuje frontu produktu IBM MQ s názvem Q1 , která patří lokálnímu správci front. Tato fronta může být lokální fronta, alias fronta nebo definice vzdálené fronty.

Metoda createQueue() může také přijmout identifikátor URI (Uniform Resource Identifier) fronty jako parametr. Identifikátor URI fronty je řetězec, který určuje název fronty produktu IBM MQ a volitelně název správce front, který je vlastníkem fronty, a jednu či více vlastností objektu JmsQueue . Následující příkaz obsahuje příklad identifikátoru URI fronty:

```
JmsQueue q2 = ff.createQueue("queue://QM2/Q2?persistence=2&priority=5");
```

Objekt JmsQueue vytvořený tímto příkazem představuje frontu IBM MQ s názvem Q2 , kterou vlastní správce front QM2, a všechny zprávy odeslané do tohoto místa určení jsou trvalé a mají prioritu 5. Další informace o identifikátorech URI front viz “Uniform resource identifiers (URI)” na stránce 193. Alternativní způsob nastavení vlastností objektu JmsQueue viz “Nastavení vlastností objektů produktu IBM MQ classes for JMS” na stránce 182.

Chcete-li vytvořit objekt Topic, může aplikace použít metodu createTopic() objektu továrny JmsFactory, jak je uvedeno v následujícím příkladu:

```
JmsTopic t1 = ff.createTopic("Sport/Football/Results");
```

Tento příkaz vytvoří objekt JmsTopic s výchozími hodnotami pro všechny jeho vlastnosti. Objekt představuje téma s názvem Sport/Football/Results.

Metoda createTopic() může také přijmout URI tématu jako parametr. Identifikátor URI tématu je řetězec, který uvádí název tématu a volitelně jednu nebo více vlastností objektu JmsTopic . Následující příkazy obsahují příklad identifikátoru URI tématu:

```
String s1 = "topic://Sport/Tennis/Results?persistence=1&priority=0";
JmsTopic t2 = ff.createTopic(s1);
```

Objekt JmsTopic vytvořený těmito příkazy představuje téma s názvem Sport/Tennis/Results a všechny zprávy odeslané do tohoto místa určení jsou dočasné a mají prioritu 0. Další informace o identifikátorech URI tématu viz “Uniform resource identifiers (URI)” na stránce 193. Alternativním způsobem nastavení vlastností objektu JmsTopic naleznete v tématu “Nastavení vlastností objektů produktu IBM MQ classes for JMS” na stránce 182.

Poté, co aplikace vytvořila továrnu připojení nebo místo určení, lze tento objekt použít pouze s vybraným poskytovatelem systému zpráv.

Nastavení vlastností objektů produktu IBM MQ classes for JMS

Chcete-li nastavit vlastnosti objektů IBM MQ classes for JMS s použitím rozšíření produktu IBM JMS , aplikace použije metody rozhraní com.ibm.msg.client.JmsPropertyContext .

Pro každý datový typ produktu Java obsahuje kontextové rozhraní JmsPropertymetodu pro nastavení hodnoty vlastnosti s daným datovým typem a metodu pro získání hodnoty vlastnosti s daným datovým typem. Aplikace volá například metodu setIntProperty () k nastavení vlastnosti s celočíselnou hodnotou a volá metodu getIntProperty () k získání vlastnosti s celočíselnou hodnotou.

Instance tříd v balíku com.ibm.mq.jms také dědí metody rozhraní kontextu JmsProperty. Aplikace může proto tyto metody použít k nastavení vlastností objektů MQConnectionFactory, MQQueue a MQTopic.

Když aplikace vytvoří objekt IBM MQ classes for JMS , všechny vlastnosti s výchozími hodnotami jsou nastaveny automaticky. Když aplikace nastaví vlastnost, nová hodnota nahradí jakoukoli předchozí hodnotu, kterou vlastnost měla. Po nastavení vlastnosti nelze tuto vlastnost odstranit, ale její hodnotu lze změnit.

Pokusí-li se aplikace o nastavení vlastnosti na hodnotu, která není platnou hodnotou vlastnosti, produkt IBM MQ classes for JMS vyvolá výjimku JMSException. Pokusí-li se aplikace o získání vlastnosti, která nebyla nastavena, chování je popsáno ve specifikaci JMS . IBM MQ classes for JMS vrací výjimku NumberFormatException pro primitivní datové typy a vrací hodnotu null pro odkazované datové typy.

Kromě předdefinovaných vlastností objektu IBM MQ classes for JMS může aplikace nastavit své vlastní vlastnosti. Tyto vlastnosti definované aplikací jsou produktem IBM MQ classes for JMSignorovány.

Další informace o vlastnostech objektů produktu IBM MQ classes for JMS naleznete v tématu Vlastnosti objektů IBM MQ classes for JMS.

Následující kód je příkladem, jak nastavit vlastnosti pomocí rozšíření produktu IBM JMS . Kód nastaví pět vlastností továrny připojení.

```
factory.setIntProperty(WMQConstants.WMQ_CONNECTION_MODE,
WMQConstants.WMQ_CM_CLIENT);
factory.setStringProperty(WMQConstants.WMQ_QUEUE_MANAGER, "QM1");
factory.setStringProperty(WMQConstants.WMQ_HOST_NAME, "HOST1");
factory.setIntProperty(WMQConstants.WMQ_PORT, 1415);
factory.setStringProperty(WMQConstants.WMQ_CHANNEL, "QM1.SVR");
factory.setStringProperty(WMQConstants.WMQ_APPLICATIONNAME, "My Application");
```

Výsledkem nastavení těchto vlastností je to, že se aplikace připojuje ke správci front QM1 v režimu klienta pomocí kanálu MQI s názvem QM1.SVR. Správce front je spuštěn v systému s názvem hostitele HOST1a modul listener pro správce front naslouchá na portu číslo 1415. Toto připojení a další připojení správce front přidružená k relacím pod ním mají název aplikace "Moje aplikace" přidružená k těmto relacím.

Poznámka: Správci front spuštěným na platformách z/OS nepodporují nastavení názvů aplikací, a toto nastavení je proto ignorováno.

Kontextové rozhraní JmsPropertyobsahuje také metodu setObjectProperty (), kterou může aplikace použít k nastavení vlastností. Druhým parametrem metody je objekt, který zapouzdřuje hodnotu vlastnosti. Například následující kód vytvoří celočíselný objekt, který zapouzdří celé číslo 1415, a pak vyvolá funkci setObjectProperty () k nastavení vlastnosti PORT továrny na připojení na hodnotu 1415:

```
Integer port = new Integer(1415);
factory.setObjectProperty(WMQConstants.WMQ_PORT, port);
```

Tento kód je proto ekvivalentem následujícího příkazu:

```
factory.setIntProperty(WMQConstants.WMQ_PORT, 1415);
```

Obráceně metoda getObjectProperty () naopak vrací objekt, který zapouzdřuje hodnotu vlastnosti.

Implicitní převod hodnoty vlastnosti z jednoho datového typu na jiný.

Když aplikace používá metodu JmsPropertyKontextové rozhraní k nastavení nebo získání vlastnosti objektu IBM MQ classes for JMS , hodnota vlastnosti může být implicitně převedena z jednoho datového typu na jiný.

Následující příkaz například nastavuje vlastnost PRIORITY objektu JmsQueue q1:

```
q1.setStringProperty(WMQConstants.WMQ_PRIORITY, "5");
```

Vlastnost PRIORITY má celočíselnou hodnotu a proto volání funkce setString() implicitně převede řetězec "5" (zdrojová hodnota) na celočíselnou hodnotu 5 (cílovou hodnotu), která se pak stane hodnotou vlastnosti PRIORITY.

Naopak, následující příkaz získá vlastnost PRIORITY objektu JmsQueue q1:

```
String s1 = q1.getStringProperty(WMQConstants.WMQ_PRIORITY);
```

Celočíselná hodnota 5 (zdrojová hodnota), která je hodnotou vlastnosti PRIORITY, je implicitně převedena na řetězec "5" (cílová hodnota) voláním funkce getStringProperty ().

Převody podporované produktem IBM MQ classes for JMS jsou zobrazeny v Tabulka 33 na stránce 183.

Tabulka 33. Podporované konverze z jednoho datového typu do jiného

Zdrojový datový typ	**Podporované cílové datové typy**
typ boolean	Řetězec
bajt	int, long, short, String
ZNAK	Řetězec
dvojitý	Řetězec
float	dvojitý, String
celé číslo	dlouhý, řetězec
long	Řetězec

Tabulka 33. Podporované konverze z jednoho datového typu do jiného (pokračování)	
Zdrojový datový typ	**Podporované cílové datové typy**
short	int, long, String
Řetězec	logická hodnota, byte, double, float, int, long, short

Obecná pravidla týkající se podporovaných převodů jsou následující:

- Číselné hodnoty lze převádět z jednoho datového typu na jiný za předpokladu, že během převodu nebudou ztracena žádná data. Např. hodnota s datovým typem `int` může být převedena na hodnotu s datovým typem `long`, ale nemůže být převedena na hodnotu s datovým typem `short`.
- Hodnota libovolného datového typu může být převedena na řetězec.
- Řetězec může být převeden na hodnotu libovolného jiného datového typu (kromě `char`) za předpokladu, že řetězec je ve správném formátu pro převod. Pokusí-li se aplikace o převod řetězce, který není ve správném formátu, IBM MQ classes for JMS vyvolá výjimku NumberFormatException.
- Pokud se aplikace pokusí o převod, který není podporován, příkaz IBM MQ classes for JMS vyvolá výjimku výjimky MessageFormat.

Specifická pravidla pro převod hodnoty z jednoho datového typu do jiného jsou následující:

- Při převodu logické hodnoty na řetězec je hodnota `true` převedena na řetězec "true"a hodnota `false` se převede na řetězec "false".
- Při převodu řetězce na logickou hodnotu je řetězec "true" (bez rozlišování velkých a malých písmen) převeden na `true`a řetězec "false" (bez rozlišování velkých a malých písmen) se převede na `false`. Jakýkoli jiný řetězec se převede na `false`.
- Při převodu řetězce na hodnotu s datovým typem `byte`, `int`, `long`nebo `short`musí řetězec obsahovat následující formát:

  [*mezery*] [*znak*] *číslice*

  Význam komponent řetězce je následující:

  mezery
  Volitelné úvodní prázdné znaky.

  SIGN
  Volitelné znaménko plus (+) nebo znaménko minus (-).

  číslice
  Souvislá posloupnost číslic (0-9). Musí být přítomna alespoň jedna číslice.

  Po posloupnosti číslic může řetězec obsahovat jiné znaky, které nejsou číslice, ale konverze se zastaví, jakmile se dosáhne prvního z těchto znaků. Předpokládá se, že řetězec představuje desítkové celé číslo.

  Pokud řetězec není ve správném formátu, příkaz IBM MQ classes for JMS vyvolá výjimku výjimky NumberFormat.
- Při převodu řetězce na hodnotu s datovým typem `double` nebo `float`musí řetězec obsahovat následující formát:

  [*mezery*] [*znak*] *číslice* [*ne_znak* [*_znak*] *_číslice*]

  Význam komponent řetězce je následující:

  mezery
  Volitelné úvodní prázdné znaky.

  SIGN
  Volitelné znaménko plus (+) nebo znaménko minus (-).

  číslice
  Souvislá posloupnost číslic (0-9). Musí být přítomna alespoň jedna číslice.

_char
Znak exponent, který je buď *E* , nebo *e*.

_sign
Volitelný znak plus (+) nebo minus (-) pro exponent.

_číslice
Souvislá posloupnost číslic (0-9) pro exponent. Pokud řetězec obsahuje znak exponent, musí být přítomna alespoň jedna číslice.

Po posloupnosti číslic nebo volitelných znaků představujících exponent může řetězec obsahovat jiné znaky, které nejsou číslice, ale konverze se zastaví, jakmile se dosáhne prvního z těchto znaků. Předpokládá se, že řetězec představuje desetinné číslo s plovoucí řádovou čárkou s exponentem, který je mocninou 10.

Pokud řetězec není ve správném formátu, příkaz IBM MQ classes for JMS vyvolá výjimku výjimky NumberFormat.

- Při převodu číselné hodnoty (včetně hodnoty s datovým typem `byte`) na řetězec, hodnota se převede na řetězcovou reprezentaci hodnoty jako dekadické číslo, nikoli řetězec obsahující znak ASCII pro tuto hodnotu. Například celé číslo 65 se převede na řetězec "65", nikoli na řetězec "A".

Nastavení více než jedné vlastnosti v jednom volání

Rozhraní kontextu JmsPropertytaké obsahuje metodu setBatchProperties (), kterou může aplikace použít k nastavení více než jedné vlastnosti v rámci jednoho volání. Parametr metody je objekt mapy, který zapouzdřuje sadu dvojic název-hodnota-hodnota.

Následující kód například používá metodu setBatchProperties () k nastavení stejných pěti vlastností továrny připojení, jak je uvedeno v tématu “Nastavení vlastností objektů produktu IBM MQ classes for JMS” na stránce 182. Kód vytvoří instanci třídy HashMap , která implementuje rozhraní Map.

```
HashMap batchProperties = new HashMap();
batchProperties.put(WMQConstants.WMQ_CONNECTION_MODE,
                    new Integer(WMQConstants.WMQ_CM_CLIENT));
batchProperties.put(WMQConstants.WMQ_QUEUE_MANAGER, "QM1");
batchProperties.put(WMQConstants.WMQ_WMQ_HOST_NAME, "HOST1");
batchProperties.put(WMQConstants.WMQ_PORT, "1414");
batchProperties.put(WMQConstants.WMQ_CHANNEL, "QM1.SVR");
factory.setBatchProperties(batchProperties);
```

Všimněte si, že druhý parametr metody Map.put() musí být objekt. Proto musí být hodnota vlastnosti s primitivním datovým typem uzavřena v rámci objektu nebo znázorněna řetězcem, jak je uvedeno v příkladu.

Metoda setBatchProperties () ověřuje každou vlastnost. Pokud metoda Properties () setBatchnemůže nastavit vlastnost, protože její hodnota například není platná, není nastavena žádná z uvedených vlastností.

Názvy a hodnoty vlastností

Pokud aplikace používá metody rozhraní JmsPropertyKontextové rozhraní k nastavení a získání vlastností objektů produktu IBM MQ classes for JMS , může aplikace určit názvy a hodnoty vlastností kterýmikoli z následujících způsobů. Každý z doprovodných příkladů uvádí, jak nastavit vlastnost PRIORITY objektu JmsQueue objektu q1 tak, aby zpráva odeslaná do fronty měla prioritu určenou ve volání metody send ().

Pomocí názvů vlastností a hodnot, které jsou definovány jako konstanty v rozhraní com.ibm.msg.client.wmq.WMQConstants

Následující příkaz je příkladem toho, jak lze určit názvy a hodnoty vlastností tímto způsobem:

```
q1.setIntProperty(WMQConstants.WMQ_PRIORITY, WMQConstants.WMQ_PRI_APP);
```

Použití názvů a hodnot vlastností, které lze použít ve frontě a v uniformných identifikátorech prostředku (URI)

Následující příkaz je příkladem toho, jak lze určit názvy a hodnoty vlastností tímto způsobem:

```
q1.setIntProperty("priority", -2);
```

Do tohoto způsobu lze zadat pouze názvy a hodnoty vlastností cílů.

Použití názvů a hodnot vlastností, které jsou rozpoznány administračním nástrojem produktu IBM MQ JMS

Následující příkaz je příkladem toho, jak lze určit názvy a hodnoty vlastností tímto způsobem:

```
q1.setStringProperty("PRIORITY", "APP");
```

Zkrácený tvar názvu vlastnosti je také přijatelný, jak je uvedeno v následujícím příkazu:

```
q1.setStringProperty("PRI", "APP");
```

Když aplikace získá vlastnost, vrácená hodnota závisí na způsobu, jakým aplikace uvádí název vlastnosti. Například, pokud aplikace uvádí konstantu WMQConstants.WMQ_PRIORITY jako název vlastnosti, vrácená hodnota je celé číslo -2:

```
int n1 = getIntProperty(WMQConstants.WMQ_PRIORITY);
```

Stejná hodnota je vrácena, pokud aplikace uvádí řetězec "priority" jako název vlastnosti:

```
int n2 = getIntProperty("priority");
```

Pokud však aplikace specifikuje řetězec "PRIORITY" nebo "PRI" jako název vlastnosti, vrácená hodnota je řetězec "APP":

```
String s1 = getStringProperty("PRI");
```

Interně IBM MQ classes for JMS ukládá názvy vlastností a hodnoty jako literálové hodnoty definované v rozhraní com.ibm.msg.client.wmq.WMQConstants . Jedná se o definovaný kanonický formát pro názvy vlastností a hodnoty. Obecným pravidlem je, že pokud aplikace nastavuje vlastnosti pomocí jednoho z dalších dvou způsobů určení názvů a hodnot vlastností, produkt IBM MQ classes for JMS musí převést názvy a hodnoty z uvedeného vstupního formátu do kanonického formátu. Podobně platí, že pokud aplikace získá vlastnosti pomocí jednoho z dalších dvou způsobů zadání názvů a hodnot vlastností, produkt IBM MQ classes for JMS musí převést názvy z určeného vstupního formátu do kanonického formátu a převést hodnoty z kanonického formátu do požadovaného výstupního formátu. Provedení těchto převodů může mít vliv na výkon.

Názvy vlastností a hodnoty vrácené výjimkami, v souborech trasování nebo v protokolu IBM MQ classes for JMS jsou vždy v kanonickém formátu.

Použití rozhraní Map

Rozhraní kontextu JmsPropertyrozšiřuje rozhraní java.util.Map . Aplikace proto může používat metody rozhraní Map pro přístup k vlastnostem objektu IBM MQ classes for JMS .

Následující kód například vytiskne názvy a hodnoty všech vlastností továrny připojení. Kód používá pouze metody mapových rozhraní k získání názvů a hodnot vlastností.

```
// Get the names of all the properties
Set propNames = factory.keySet();

// Loop round all the property names and get the property values
```

```
Iterator iterator = propNames.iterator();
while (iterator.hasNext()){
    String pName = (String)iterator.next();
    System.out.println(pName+"="+factory.get(pName));
}
```

Použití metod Map interface neobejdete žádné ověření platnosti vlastností ani konverze.

Použití rozšíření produktu IBM MQ JMS
IBM MQ classes for JMS obsahuje sadu rozšíření rozhraní API JMS s názvem rozšíření IBM MQ JMS . Aplikace může tato rozšíření použít k dynamickému vytváření továren připojení a cílů v běhovém prostředí a k nastavení vlastností továren připojení a cílů.

IBM MQ classes for JMS obsahuje sadu tříd v balících com.ibm.jms a com.ibm.mq.jms. Tyto třídy implementují rozhraní JMS a obsahují rozšíření IBM MQ JMS . Příklady kódu, které následují za předpokladu, že tyto balíky byly importovány následujícími příkazy:

```
import com.ibm.jms.*;
import com.ibm.mq.jms.*;
import com.ibm.msg.client.wmq.WMQConstants;
```

Aplikace může používat rozšíření produktu IBM MQ JMS k provádění následujících funkcí:

- Dynamicky vytvářet továrny připojení a místa určení za běhu namísto jejich načtení jako spravovaných objektů z oboru názvů rozhraní JNDI (Java Naming and Directory Interface)
- Nastavit vlastnosti továren připojení a cílů

Vytváření továren na připojení

Chcete-li vytvořit továrnu na připojení, může aplikace použít konstruktor MQConnectionFactory , jak ukazuje následující příklad:

```
MQConnectionFactory factory = new MQConnectionFactory();
```

Tento příkaz vytvoří objekt MQConnectionFactory s výchozími hodnotami pro všechny jeho vlastnosti, což znamená, že se aplikace připojí k výchozímu správci front v režimu vazeb. Chcete-li, aby se aplikace připojovala v režimu klienta nebo se připojila k jinému správci front, než je výchozí správce front, musí aplikace před vytvořením připojení nastavit odpovídající vlastnosti objektu MQConnectionFactory . Další informace o tom, jak to provést, viz “Nastavení vlastností továren připojení” na stránce 187.

Aplikace může vytvořit továrny připojení následujících typů podobným způsobem:

- Továrna MQQueueConnection
- Továrna MQTopicConnection
- MQXAConnectionFactory
- Továrna MQXAQueueConnection
- Továrna MQXATopicConnection

Nastavení vlastností továren připojení

Aplikace může nastavit vlastnosti továrny připojení voláním příslušných metod továrny připojení. Továrnu připojení může buď být spravovaný objekt, nebo objekt vytvořený dynamicky za běhu.

Prohlédněte si následující kód, například:

```
MQConnectionFactory factory = new MQConnectionFactory();
.
factory.setTransportType(WMQConstants.WMQ_CM_CLIENT);
factory.setQueueManager("QM1");
factory.setHostName("HOST1");
```

```
factory.setPort(1415);
factory.setChannel("QM1.SVR");
```

Tento kód vytvoří objekt MQConnectionFactory a poté nastaví pět vlastností objektu. Výsledkem nastavení těchto vlastností je to, že se aplikace připojuje ke správci front QM1 v režimu klienta pomocí kanálu MQI s názvem QM1.SVR. Správce front je spuštěn v systému s názvem hostitele HOST1a modul listener pro správce front naslouchá na portu číslo 1415.

V případě připojení v reálném čase ke zprostředkovateli může aplikace použít následující kód:

```
MQConnectionFactory factory = new MQConnectionFactory();
.
factory.setTransportType(WMQConstants.WMQ_CM_DIRECT);
factory.setHostName("HOST2");
factory.setPort(1507);
```

Tento kód předpokládá, že zprostředkovatel běží na systému s názvem hostitele HOST2 a naslouchá na portu číslo 1507.

Aplikace, která používá v reálném čase připojení k zprostředkovateli, může používat pouze styl publikování/odběru zpráv. Nemůže používat styl systému zpráv typu point-to-point.

Platné jsou pouze určité kombinace vlastností továrny připojení. Informace o tom, které kombinace jsou platné, najdete v tématu Závislosti mezi vlastnostmi objektů produktu IBM MQ classes for JMS.

Další informace o vlastnostech továrny připojení a metodách používaných k nastavení jejích vlastností naleznete v tématu Vlastnosti objektů IBM MQ classes for JMS.

Vytváření cílů

Chcete-li vytvořit objekt fronty, může aplikace použít konstruktor MQQueue, jak je uvedeno v následujícím příkladu:

```
MQQueue q1 = new MQQueue("Q1");
```

Tento příkaz vytvoří objekt MQQueue s výchozími hodnotami pro všechny jeho vlastnosti. Objekt představuje frontu produktu IBM MQ s názvem Q1 , která patří lokálnímu správci front. Tato fronta může být lokální fronta, alias fronta nebo definice vzdálené fronty.

Alternativní formulář konstruktoru MQQueue má dva parametry, jak je zobrazeno v následujícím příkladu:

```
MQQueue q2 = new MQQueue("QM2", "Q2");
```

Objekt MQQueue vytvořený tímto příkazem reprezentuje frontu produktu IBM MQ s názvem Q2 , kterou vlastní správce front QM2. Identifikovaným správcem front může být lokální správce front nebo vzdálený správce front. Pokud se jedná o vzdáleného správce front, musí být produkt IBM MQ nakonfigurován tak, aby v případě, že aplikace odešle zprávu do tohoto místa určení, mohla produkt WebSphere MQ směrovat zprávu z lokálního správce front do vzdáleného správce front.

Konstruktor MQQueue může také přijmout identifikátor URI (Uniform Resource Identifier) fronty jako jeden parametr. Identifikátor URI fronty je řetězec, který určuje název fronty produktu IBM MQ a volitelně název správce front, který je vlastníkem fronty, a jednu či více vlastností objektu MQQueue. Následující příkaz obsahuje příklad identifikátoru URI fronty:

```
MQQueue q3 = new MQQueue("queue://QM3/Q3?persistence=2&priority=5");
```

Objekt MQQueue vytvořený tímto příkazem reprezentuje frontu IBM MQ s názvem Q3 , která je vlastněna správcem front QM3, a všechny zprávy odeslané do tohoto místa určení jsou trvalé a mají prioritu 5. Další informace o identifikátorech URI front viz “Uniform resource identifiers (URI)” na stránce 193. Alternativním způsobem nastavení vlastností objektu MQQueue naleznete v tématu “Nastavení vlastností cílů” na stránce 189.

Chcete-li vytvořit objekt Topic, může aplikace použít konstruktor MQTopic, jak je uvedeno v následujícím příkladu:

```
MQTopic t1 = new MQTopic("Sport/Football/Results");
```

Tento příkaz vytvoří objekt MQTopic s výchozími hodnotami pro všechny jeho vlastnosti. Objekt představuje téma s názvem Sport/Football/Results.

Konstruktor MQTopic může také přijmout identifikátor URI tématu jako parametr. Identifikátor URI tématu je řetězec, který určuje název tématu a volitelně také jednu nebo více vlastností objektu MQTopic. Následující příkaz obsahuje příklad identifikátoru URI tématu:

```
MQTopic t2 = new MQTopic("topic://Sport/Tennis/Results?persistence=1&priority=0");
```

Objekt MQTopic vytvořený tímto příkazem představuje téma s názvem Sport/Tennis/Results a všechny zprávy odeslané do tohoto místa určení jsou dočasné a mají prioritu 0. Další informace o identifikátorech URI tématu viz "Uniform resource identifiers (URI)" na stránce 193. Alternativním způsobem nastavení vlastností objektu MQTopic naleznete v tématu "Nastavení vlastností cílů" na stránce 189.

Nastavení vlastností cílů

Aplikace může nastavit vlastnosti cíle voláním odpovídajících metod cíle. Místo určení může být buď administrovaný objekt, nebo objekt vytvořený dynamicky za běhu.

Prohlédněte si následující kód, například:

```
MQQueue q1 = new MQQueue("Q1");
.
q1.setPersistence(WMQConstants.WMQ_PER_PER);
q1.setPriority(5);
```

Tento kód vytvoří objekt MQQueue a poté nastaví dvě vlastnosti objektu. Výsledkem nastavení těchto vlastností je to, že všechny zprávy odeslané do místa určení jsou trvalé a mají prioritu 5.

Aplikace může nastavit vlastnosti objektu MQTopic podobným způsobem, jak je uvedeno v následujícím příkladu:

```
MQTopic t1 = new MQTopic("Sport/Football/Results");
.
t1.setPersistence(WMQConstants.WMQ_PER_NON);
t1.setPriority(0);
```

Tento kód vytvoří objekt MQTopic a poté nastaví dvě vlastnosti objektu. Výsledkem nastavení těchto vlastností je, že všechny zprávy odeslané do místa určení jsou netrvalé a mají prioritu 0.

Další informace o vlastnostech cíle a metodách použitých k nastavení jejích vlastností naleznete v tématu Vlastnosti objektů IBM MQ classes for JMS.

Sestavení připojení v aplikaci JMS

Chcete-li vytvořit připojení, aplikace JMS použije objekt ConnectionFactory k vytvoření objektu připojení a pak spustí připojení.

Chcete-li vytvořit objekt připojení, aplikace použije metodu createConnection() objektu ConnectionFactory , jak je uvedeno v následujícím příkladu:

```
ConnectionFactory factory;
Connection connection;
.
.
.
connection = factory.createConnection();
```

Je-li vytvořeno připojení k produktu JMS , produkt IBM MQ classes for JMS vytvoří popisovač připojení (Hconn) a zahájí konverzaci se správcem front.

Rozhraní továrny QueueConnectiona rozhraní továrny TopicConnectioneach dědí metodu createConnection() z rozhraní ConnectionFactory . Proto můžete použít metodu createConnection() k vytvoření objektu specifického pro doménu, jak je zobrazeno v následujícím příkladu:

```
QueueConnectionFactory qcf;
Connection connection;
.
.
.
connection = qcf.createConnection();
```

Tento fragment kódu vytvoří objekt QueueConnection . Aplikace může nyní na tomto objektu provést nezávislou operaci na doméně, nebo operaci, která je použitelná pouze pro doménu typu point-to-point. Pokud se však aplikace pokusí provést operaci, která je použitelná pouze pro doménu publikování/odběru, vyvolá se výjimka výjimky IllegalStates touto zprávou:

```
JMSMQ1112: Operation for a domain specific object was not valid.
           Operation createProducer() is not valid for type com.ibm.mq.jms.MQTopic
```

Důvodem je to, že připojení bylo vytvořeno z továrny připojení specifické pro doménu.

Poznámka: Všimněte si, že ID procesu aplikace se používá jako výchozí identita uživatele, která má být předána správci front. Je-li aplikace spuštěna v režimu transportu klienta, musí toto ID procesu existovat spolu s příslušnými oprávněními na serveru. Chcete-li použít jinou identitu, použijte metodu createConnection(jméno uživatele, heslo).

Ve specifikaci JMS je uvedeno, že připojení je vytvořeno ve stavu `stopped` . Do zahájení připojení nemůže spotřebitel zpráv, který je přidružen k tomuto připojení, přijímat žádné zprávy. Chcete-li spustit připojení, aplikace použije metodu start () objektu připojení, jak je zobrazeno v následujícím příkladu:

```
connection.start();
```

Vytvoření relace v aplikaci JMS

K vytvoření relace používá aplikace JMS metodu createSession() objektu Connection.

Metoda createSession() má dva parametry:

1. Parametr, který určuje, zda relace obsahuje transakci, nebo nikoli.
2. Parametr, který uvádí režim potvrzení pro relaci

Například následující kód vytvoří relaci, která se neobchoduje a má režim potvrzení AUTO_ACKNOWLEDGE:

```
Session session;
.
boolean transacted = false;
session = connection.createSession(transacted, Session.AUTO_ACKNOWLEDGE);
```

Když je vytvořena relace JMS , vytvoří IBM MQ classes for JMS popisovač připojení (Hconn) a spustí konverzaci se správcem front.

Objekt relace a objekt MessageProducer nebo MessageConsumer z něj vytvořené nemohou být souběžně používány různými podprocesy aplikace s podporou podprocesů. Nejjednodušším způsobem, jak zajistit, aby tyto objekty nebyly použity souběžně, je vytvoření samostatného objektu relace pro každý podproces.

Překonat relace v aplikacích produktu JMS

Aplikace produktu JMS mohou spouštět lokální transakce při prvním vytvoření relace transakce. Aplikace může potvrdit nebo odvolat transakci.

Aplikace produktu JMS mohou spouštět lokální transakce. Lokální transakce je transakce, která zahrnuje změny pouze na prostředky správce front, ke kterému je aplikace připojena. Chcete-li spustit lokální transakce, musí aplikace nejprve vytvořit relaci transakce voláním metody createSession() objektu připojení, která určuje jako parametr, že relace je zpracovávána. Následně jsou všechny zprávy odeslané a přijaté v rámci relace seskupeny do posloupnosti transakcí. Transakce je ukončena, když aplikace potvrdí nebo odvolá zprávy, které odeslala a obdržela od začátku transakce.

Pro potvrzení transakce volá aplikace metodu commit () objektu Session. Když je transakce potvrzena, všechny zprávy odeslané v rámci transakce se stanou dostupnými pro doručení do jiných aplikací a všechny zprávy přijaté v rámci transakce jsou potvrzeny, aby se server systému zpráv nepokoušel o jejich doručení do aplikace znovu. V doméně typu point-to-point server zpráv také odebírá přijaté zprávy z jejich front.

Chcete-li odvolat transakci, volá aplikace metodu rollback () objektu Session. Když je transakce odvolána, všechny zprávy odeslané v rámci transakce jsou vyřazeny serverem systému zpráv a všechny zprávy přijaté v rámci transakce budou znovu k dispozici pro doručení. V dvoubodové doméně se zprávy, které byly přijaty, vrátí zpět do front a znovu se stanou viditelnými pro jiné aplikace.

Nová transakce se spustí automaticky, když aplikace vytvoří transakci s transakcemi nebo zavolá metodu commit () nebo rollback (). Proto má relace, která obsahuje transakci, vždy aktivní transakci.

Když aplikace zavře relaci transakce, dojde k implicitnímu odvolání transakce. Když aplikace uzavře připojení, dojde k implicitnímu odvolání transakce pro všechny relace, které se v relaci nachází.

Pokud aplikace skončí bez zavření připojení, dojde také k implicitnímu odvolání transakce pro všechny relace obsažené v transakci, které se týká.

Transakce je zcela obsažena v rámci relace transakce. Transakce nemůže přesahovat relace. To znamená, že aplikace nemůže odesílat a přijímat zprávy ve dvou nebo více transakčních relacích, a poté potvrdit nebo odvolat všechny tyto akce jako jedinou transakci.

Režimy potvrzení relací JMS
Každá relace, která nemá transakci, má režim potvrzení, který určuje, jak jsou přijímány zprávy přijaté aplikací. K dispozici jsou tři režimy potvrzení a volba režimu potvrzení má vliv na návrh aplikace.

Pokud relace není součástí transakce, je způsob, jakým zprávy přijaté aplikací potvrdí, je určen režimem potvrzení relace. Tři režimy potvrzení jsou popsány v následujících odstavcích:

AUTOMATICKÉ_POTVRZENÍ

Relace automaticky potvrdí každou zprávu přijatou aplikací.

Pokud jsou zprávy doručovány synchronně do aplikace, relace potvrdí příjem zprávy pokaždé, když je úspěšně dokončeno přijetí volání. Pokud jsou zprávy doručovány asynchronně, relace potvrdí přijetí zprávy pokaždé, když se úspěšně dokončí volání metody onMessage() modulu listener pro zprávy.

Pokud aplikace obdrží zprávu úspěšně, ale selhání zabrání provedení potvrzení, bude zpráva znovu k dispozici pro doručení. Aplikace musí proto být schopna zpracovat znovu dodanou zprávu.

PUP_OK_ACKNOWLEDGE

Relace bere na vědomí zprávy přijaté aplikací v době, kdy je vybrána.

Použití tohoto režimu potvrzení snižuje množství práce, které musí relace dělat, ale selhání zabraňující potvrzení zprávy může vést k tomu, že bude znovu k dispozici více než jedna zpráva k doručení. Aplikace musí být proto schopna zpracovávat zprávy, které jsou znovu doručeny.

Omezení: V režimech AUTO_ACKNOWLEDGE a DUPS_OK_ACKNOWLEDGE produkt JMS nepodporuje aplikaci zahození neošetřené výjimky v modulu listener pro zprávy. To znamená, že zprávy jsou vždy potvrzeny, když se modul listener pro zprávy vrátí, bez ohledu na to, zda byl úspěšně zpracován (za předpokladu, že došlo k neúspěchům selhání) a nezabráníte tomu, aby aplikace pokračovala). Pokud vyžadujete lepší kontrolu potvrzení zprávy, použijte režimy CLIENT_ACKNOWLEDGE nebo transakci, které poskytují aplikaci plnou kontrolu nad funkcemi potvrzení.

CLIENT_ACKNOWLEDGE

Aplikace potvrdí zprávy, které obdrží, voláním metody Acknowledge třídy Message.

Aplikace může potvrdit příjem každé zprávy jednotlivě, nebo může obdržet dávku zpráv a zavolat metodu Potvrdit pouze pro poslední zprávu, kterou obdrží. Když se metoda Acknowledge nazývá všechny zprávy přijaté od poslední doby, kdy byla metoda volána, jsou potvrzeny.

Ve spojení s jakýmkoli z těchto režimů potvrzení může aplikace zastavit a znovu spustit doručování zpráv v relaci voláním metody Recover třídy Session. Přijaté zprávy, ale dříve nepotvrzené, jsou znovu doručeny. Je však možné, že nebudou doručeny ve stejné posloupnosti, v jaké byly dříve doručeny. Do té doby mohla dorazila zpráva s vyšší prioritou a některé původní zprávy mohly vypršet. V doméně dvoubodového spojení mohly být některé z původních zpráv spotřebovány jinou aplikací.

Aplikace může určit, zda je zpráva znovu doručena, tím, že zkontrolujete obsah pole záhlaví JMSRedelivered zprávy. Aplikace to provede voláním metody getJMSRedelivered() třídy zpráv.

Vytvoření cílů v aplikaci JMS

Místo načítání cílů jako spravovaných objektů z oboru názvů rozhraní JNDI (Java Naming and Directory Interface) může aplikace produktu JMS použít relaci k dynamickému vytváření cílů v době běhu programu. Aplikace může použít identifikátor URI (Uniform Resource Identifier) k identifikaci fronty produktu IBM MQ nebo tématu a volitelně také k určení jedné či více vlastností objektu Fronta nebo Téma.

Použití relace k vytvoření objektů fronty

Chcete-li vytvořit objekt fronty, může aplikace použít metodu createQueue() objektu relace, jak je uvedeno v následujícím příkladu:

```
Session session;
.
Queue q1 = session.createQueue("Q1");
```

Tento kód vytvoří objekt fronty s výchozími hodnotami pro všechny jeho vlastnosti. Objekt představuje frontu produktu IBM MQ s názvem Q1 , která patří lokálnímu správci front. Tato fronta může být lokální fronta, alias fronta nebo definice vzdálené fronty.

Metoda createQueue() také přijímá identifikátor URI fronty jako parametr. Identifikátor URI fronty je řetězec, který určuje název fronty produktu IBM MQ a volitelně název správce front, který vlastní frontu, a jednu či více vlastností objektu Fronta. Následující příkaz obsahuje příklad identifikátoru URI fronty:

```
Queue q2 = session.createQueue("queue://QM2/Q2?persistence=2&priority=5");
```

Objekt fronty vytvořený tímto příkazem reprezentuje frontu IBM MQ s názvem Q2 , kterou vlastní správce front s názvem QM2, a všechny zprávy odeslané do tohoto místa určení jsou trvalé a mají prioritu 5. Identifikovaným správcem front může být lokální správce front nebo vzdálený správce front. Pokud se jedná o vzdáleného správce front, musí být produkt IBM MQ nakonfigurován tak, aby v případě, že aplikace odešle zprávu do tohoto místa určení, mohla produkt WebSphere MQ směrovat zprávu z lokálního správce front do správce front QM2. Další informace o identifikátorech URI viz “Uniform resource identifiers (URI)” na stránce 193.

Všimněte si, že parametr na metodě createQueue() obsahuje informace specifické pro poskytovatele. Proto při použití metody createQueue() pro vytvoření objektu Queue místo načtení objektu Queue jako spravovaného objektu z oboru názvů rozhraní JNDI může být vaše aplikace méně přenosná.

Aplikace může vytvořit objekt TemporaryQueue pomocí metody createTemporaryQueue () objektu Session, jak je uvedeno v následujícím příkladu:

```
TemporaryQueue q3 = session.createTemporaryQueue();
```

Přestože se relace používá k vytvoření dočasné fronty, rozsah dočasné fronty je připojení, které bylo použito k vytvoření relace. Kterákoli z relací připojení může vytvořit správce zpráv a spotřebitele zpráv

pro dočasnou frontu. Dočasná fronta zůstane až do konce připojení nebo aplikace explicitně odstraní dočasnou frontu pomocí metody .delete () TemporaryQueue.delete (), podle toho, co nastane dříve.

Když aplikace vytvoří dočasnou frontu, produkt IBM MQ classes for JMS vytvoří dynamickou frontu ve správci front, ke kterému je aplikace připojena. Vlastnost TEMPMODEL továrny připojení uvádí název modelové fronty, která se používá k vytvoření dynamické fronty, a vlastnost TEMPQPREFIX továrny připojení uvádí předponu, která se používá k vytvoření názvu dynamické fronty.

Použití relace k vytvoření objektů tématu

Chcete-li vytvořit objekt Topic, může aplikace použít metodu createTopic() objektu relace, jak je uvedeno v následujícím příkladu:

```
Session session;
.
Topic t1 = session.createTopic("Sport/Football/Results");
```

Tento kód vytvoří objekt tématu s výchozími hodnotami pro všechny jeho vlastnosti. Objekt představuje téma s názvem Sport/Football/Results.

Metoda createTopic() také přijímá identifikátor URI tématu jako parametr. Identifikátor URI tématu je řetězec, který určuje název tématu a volitelně také jednu nebo více vlastností objektu tématu. Následující kód obsahuje příklad identifikátoru URI tématu:

```
String uri = "topic://Sport/Tennis/Results?persistence=1&priority=0";
Topic t2 = session.createTopic(uri);
```

Objekt Topic vytvořený tímto kódem představuje téma s názvem Sport/Tennis/Results a všechny zprávy odeslané do tohoto místa určení jsou dočasné a mají prioritu 0. Další informace o identifikátorech URI tématu viz “Uniform resource identifiers (URI)” na stránce 193.

Všimněte si, že parametr na metodě createTopic() obsahuje informace specifické pro poskytovatele. Proto při použití metody createTopic() k vytvoření objektu Topic místo načtení objektu tématu jako spravovaného objektu z oboru názvů JNDI může být vaše aplikace méně přenosná.

Aplikace může vytvořit objekt TemporaryTopic pomocí metody Topic () objektu relace createTemporary, jak je uvedeno v následujícím příkladu:

```
TemporaryTopic t3 = session.createTemporaryTopic();
```

Ačkoli se relace používá k vytvoření dočasného tématu, rozsah dočasného tématu je připojení, které bylo použito k vytvoření relace. Kterákoli z relací připojení může vytvořit správce zpráv a spotřebitele zpráv pro dočasné téma. Dočasné téma zůstane až do konce připojení nebo aplikace explicitně odstraní dočasné téma pomocí metody .delete () TemporaryTopic.delete (), podle toho, co nastane dříve.

Když aplikace vytvoří dočasné téma, produkt IBM MQ classes for JMS vytvoří téma se jménem, které začíná se znaky TEMP/*tempTopicPrefix*, kde *tempTopicPrefix* je hodnota vlastnosti TEMPTOPICPREFIX továrny připojení.

Uniform resource identifiers (URI)

Identifikátor URI fronty je řetězec, který určuje název fronty produktu IBM MQ a volitelně název správce front, který je vlastníkem fronty, a jednu či více vlastností objektu Queue vytvořeného aplikací. Identifikátor URI tématu je řetězec, který určuje název tématu a volitelně také jednu nebo více vlastností objektu tématu vytvořeného aplikací.

Identifikátor URI fronty má následující formát:

```
queue://[ qMgrName ]/qName [? propertyName1 = propertyValue1
& propertyName2 = propertyValue2
&...]
```

Identifikátor URI tématu má následující formát:

```
topic://topicName [? propertyName1 = propertyValue1
& propertyName2 = propertyValue2
&...]
```

Proměnné v těchto formátech mají následující význam:

QmgrName
Název správce front, který vlastní frontu určenou identifikátorem URI.

Správce front může být lokálním správcem front nebo vzdáleným správcem front. Pokud se jedná o vzdáleného správce front, musí být produkt IBM MQ nakonfigurován tak, aby při odeslání zprávy do fronty produkt WebSphere MQ mohl směrovat zprávu z lokálního správce front do vzdáleného správce front.

Není-li zadán žádný název, předpokládá se lokální správce front.

qName
Název fronty produktu IBM MQ .

Fronta může být lokální fronta, alias fronta nebo definice vzdálené fronty.

Pravidla pro vytváření názvů front naleznete v tématu Pravidla pojmenování objektů produktu IBM MQ.

topicName
Název tématu.

Pravidla pro vytváření názvů témat najdete v tématu Pravidla pojmenování objektů produktu IBM MQ. Vyvarujte se použití zástupných znaků +, #, *, a? v názvech témat. Názvy témat obsahující tyto znaky mohou způsobit neočekávané výsledky při jejich přihlášení k odběru. Viz Použití řetězců témat.

***propertyName1*, *propertyName2*, ...**
Názvy vlastností objektu Queue nebo Topic vytvořeného aplikací. Produkt Tabulka 34 na stránce 194 vypíše platné názvy vlastností, které lze použít v identifikátoru URI.

Nejsou-li zadány žádné vlastnosti, má objekt Fronta nebo Téma výchozí hodnoty pro všechny své vlastnosti.

***propertyValue1*, *propertyValue2*, ...**
Hodnoty vlastností objektu Queue nebo Topic vytvořeného aplikací. Tabulka 34 na stránce 194 zobrazuje seznam platných hodnot vlastností, které lze použít v identifikátoru URI.

Hranaté závorky ([]) označují volitelnou komponentu a tři tečky (...) znamenají, že seznam dvojic název-hodnota, je-li přítomen, může obsahovat jednu nebo více dvojic název-hodnota.

Produkt Tabulka 34 na stránce 194 obsahuje seznam platných názvů vlastností a platných hodnot, které lze použít ve frontě a identifikátorech URI témat. Ačkoli administrativní nástroj produktu IBM MQ JMS používá symbolické konstanty pro hodnoty vlastností, identifikátory URI nemohou obsahovat symbolické konstanty.

Tabulka 34. Názvy vlastností a platné hodnoty pro použití ve frontě a identifikátorech URI témat

Název vlastnosti	Popis	Platné hodnoty
CCSID	Způsob reprezentace znakových dat v těle zprávy při předání zprávy do místa určení produktem IBM MQ classes for JMS .	• Jakýkoli identifikátor kódované znakové sady podporovaný produktem IBM MQ.
kódování	Jak jsou číselná data v těle zprávy představována, když produkt IBM MQ classes for JMS předá zprávu do cíle	• Libovolná platná hodnota pro pole *Kódování* v deskriptoru zpráv produktu IBM MQ .

Tabulka 34. Názvy vlastností a platné hodnoty pro použití ve frontě a identifikátorech URI témat (pokračování)

Název vlastnosti	Popis	Platné hodnoty
Vypršení	Doba platnosti pro zprávy odeslané do místa určení	• -2-Jak je uvedeno ve volání send () nebo, pokud není uveden ve volání send (), výchozí doba pro život producenta zpráv. • 0-Platnost zprávy odeslané do místa určení nikdy nevyprší. • Kladné celé číslo určující dobu života v milisekundách.
výběrové vysílání	Nastavení výběrového vysílání pro téma při použití připojení v reálném čase ke zprostředkovateli	Platné hodnoty jsou uvedeny v následujícím seznamu. Přiřazeno ke každé hodnotě je odpovídající hodnota vlastnosti MULTICAST, jak je použita v nástroji pro administraci produktu IBM MQ JMS . Popis vlastnosti MULTICAST a jeho platných hodnot naleznete v tématu Vlastnosti objektů IBM MQ classes for JMS. • -1-ASCF • 0 - zakázáno • 3-NOTR • 5-SPOLEHLIVÉ • 7-POVOLENO
trvání, perzistence	Perzistence zpráv odeslaných do místa určení	• -2-Jak je uvedeno ve volání send () nebo, pokud není uveden ve volání send (), výchozí trvání producenta zpráv. • -1-Jak je uvedeno atributem *DefPersistence* fronty nebo tématu IBM MQ . • 1-Netrvalý. • 2-Trvalý. • 3-Ekvivalentní k hodnotě HIGH pro vlastnost PERSISTENCE, jak je použita v nástroji pro administraci produktu IBM MQ JMS . Vysvětlení této hodnoty najdete v tématu “Trvalé zprávy produktu JMS” na stránce 218.

Tabulka 34. Názvy vlastností a platné hodnoty pro použití ve frontě a identifikátorech URI témat (pokračování)

Název vlastnosti	Popis	Platné hodnoty
priority (priorita)	Priorita zpráv odeslaných do místa určení	• -2-Jak je uvedeno ve volání send () nebo, pokud není uvedeno ve volání send (), výchozí priorita producentu zprávy. • -1-Jak je uvedeno v atributu *DefPriority* fronty nebo tématu IBM MQ . • Celé číslo v rozsahu 0-9 určující prioritu zpráv odeslaných do místa určení.
targetClient	Určuje, zda zprávy odeslané do místa určení obsahují záhlaví MQRFH2 .	• 0-Zprávy obsahují záhlaví MQRFH2 . • 1-Zprávy neobsahují záhlaví MQRFH2 .

Příklad: Následující identifikátor URI identifikuje frontu IBM MQ s názvem Q1 , kterou vlastní lokální správce front. Objekt fronty vytvořený s použitím tohoto identifikátoru URI má výchozí hodnoty pro všechny jeho vlastnosti.

```
queue:///Q1
```

Následující identifikátor URI identifikuje frontu produktu IBM MQ s názvem Q2 , kterou vlastní správce front s názvem QM2. Všechny zprávy odeslané do tohoto místa určení mají prioritu 6. Zbývající vlastnosti objektu Queue vytvořeného pomocí tohoto identifikátoru URI mají své výchozí hodnoty.

```
queue://QM2/Q2?priority=6
```

Následující identifikátor URI identifikuje téma s názvem Sport/Athletics/Results. Všechny zprávy odeslané do tohoto místa určení jsou netrvalé a mají prioritu 0. Zbývající vlastnosti objektu Topic vytvořeného pomocí tohoto identifikátoru URI mají své výchozí hodnoty.

```
topic://Sport/Athletics/Results?persistence=1&priority=0
```

Odesílání zpráv v aplikaci produktu JMS

Než bude moci aplikace produktu JMS odesílat zprávy do místa určení, musí nejprve vytvořit objekt MessageProducer pro místo určení. Chcete-li odeslat zprávu do místa určení, aplikace vytvoří objekt Message a pak volá metodu send () objektu MessageProducer .

Aplikace používá objekt MessageProducer k odesílání zpráv. Aplikace obvykle vytvoří objekt MessageProducer pro určité místo určení, což může být fronta nebo téma, takže všechny zprávy odeslané s producentem zpráv budou odeslány do stejného cíle. Proto, než aplikace může vytvořit objekt MessageProducer , musí nejprve vytvořit objekt Fronta nebo Téma. Informace o tom, jak vytvořit objekt Fronta nebo Téma, najdete v následujících tématech:

- “Použití JNDI k načtení spravovaných objektů v aplikaci JMS” na stránce 179
- “Použití rozšíření produktu IBM JMS” na stránce 180
- “Použití rozšíření produktu IBM MQ JMS” na stránce 187
- “Vytvoření cílů v aplikaci JMS” na stránce 192

Chcete-li vytvořit objekt MessageProducer , aplikace použije metodu createProducer() objektu relace, jak je uvedeno v následujícím příkladu:

```
MessageProducer producer = session.createProducer(destination);
```

Parametr `destination` je objekt Queue nebo Topic, který aplikace vytvořila dříve.

Dříve než může aplikace odeslat zprávu, musí vytvořit objekt Zpráva. Tělo zprávy obsahuje data aplikace a produkt JMS definuje pět typů těla zprávy:

- Bajtů
- Mapa
- Objekt
- Proud
- Text

Každý typ těla zprávy má své vlastní rozhraní JMS , které je podrozhraním rozhraní zpráv, a metoda v rozhraní relace pro vytvoření zprávy s daným typem těla. Například rozhraní pro textovou zprávu se nazývá TextMessagea aplikace používá metodu createTextMessage () objektu relace k vytvoření textové zprávy, jak je uvedeno v následujícím příkazu:

```
TextMessage outMessage = session.createTextMessage(outString);
```

Další informace o zprávách a tělech zpráv najdete v tématu “Zprávy produktu JMS” na stránce 121.

Chcete-li odeslat zprávu, aplikace použije metodu send () objektu MessageProducer , jak je zobrazeno v následujícím příkladu:

```
producer.send(outMessage);
```

Aplikace může používat metodu send () k odesílání zpráv v doméně systému zpráv. Povaha místa určení určuje, která doména systému zpráv se používá. Avšak, TopicPublisher, podřízené rozhraní MessageProducer , který je specifický pro doménu publikování/odběru, má také metodu publish (), kterou lze použít místo metody send (). Tyto dvě metody jsou funkčně stejné.

Aplikace může vytvořit objekt MessageProducer , který nemá žádné zadané místo určení. V takovém případě musí aplikace určit místo určení při volání metody send ().

Pokud aplikace odešle zprávu v rámci transakce, zpráva nebude doručena do místa určení, dokud nebude transakce potvrzena. To znamená, že aplikace nemůže odeslat zprávu a přijmout odpověď na zprávu v rámci stejné transakce.

Místo určení lze konfigurovat tak, aby při odeslání zpráv do aplikace produkt IBM MQ classes for JMS odeslal zprávu a vrátil řízení zpět aplikaci bez určení, zda správce front zprávu obdržel bezpečně. Toto je někdy označováno jako *asynchronní vložení*. Další informace viz “Asynchronní vkládání zpráv do produktu IBM MQ classes for JMS” na stránce 286.

Příjem zpráv v aplikaci JMS

Aplikace používá pro příjem zpráv spotřebitele zpráv. Odběratel trvalého tématu je spotřebitel zpráv, který přijímá všechny zprávy odeslané do místa určení, včetně těch, které byly odeslány, zatímco odběratel je neaktivní. Aplikace může vybrat zprávy, které chce přijímat, pomocí selektoru zpráv, a může přijímat zprávy asynchronně pomocí modulu listener pro zprávy.

Aplikace používá objekt MessageConsumer k příjmu zpráv. Aplikace vytvoří objekt MessageConsumer pro specifické místo určení, což může být fronta nebo téma, takže všechny zprávy přijaté s použitím spotřebitele zpráv budou přijaty ze stejného cíle. Dříve než aplikace může vytvořit objekt MessageConsumer , musí nejprve vytvořit objekt Fronta nebo Téma. Informace o tom, jak vytvořit objekt Fronta nebo Téma, najdete v následujících tématech:

- “Použití JNDI k načtení spravovaných objektů v aplikaci JMS” na stránce 179
- “Použití rozšíření produktu IBM JMS” na stránce 180
- “Použití rozšíření produktu IBM MQ JMS” na stránce 187

- “Vytvoření cílů v aplikaci JMS” na stránce 192

Chcete-li vytvořit objekt MessageConsumer , aplikace použije metodu createConsumer() objektu relace, jak je uvedeno v následujícím příkladu:

```
MessageConsumer consumer = session.createConsumer(destination);
```

Parametr `destination` je objekt Queue nebo Topic, který aplikace vytvořila dříve.

Aplikace potom použije metodu receive () objektu MessageConsumer k přijetí zprávy z místa určení, jak je zobrazeno v následujícím příkladu:

```
Message inMessage = consumer.receive(1000);
```

Parametr ve volání metody receive () uvádí, jak dlouho v milisekundách metoda čeká na příchod vyhovující zprávy, pokud není okamžitě k dispozici žádná zpráva. Vynecháte-li tento parametr, budou bloky volání po dobu neurčitou do té doby, než dorazí vhodná zpráva. Nechcete-li, aby aplikace čekala na zprávu, použijte místo toho metodu wait () receiveNoWait ().

Metoda receive () vrací zprávu určitého typu. Když například aplikace obdrží textovou zprávu, objekt vrácený voláním receive () je objekt TextMessage .

Avšak deklarovaný typ objektu vrácený voláním receive () je objekt zprávy. Proto, aby bylo možné extrahovat data z těla zprávy, která byla právě přijata, musí aplikace přetypovat ze třídy Message na specifičtější podtřídu, jako např. TextMessage. Není-li typ zprávy znám, může aplikace použít operátor `instanceof` k určení typu zprávy. Pro aplikaci je vždy dobrým zvykem určit typ zprávy před odléváním, aby bylo možné s chybami zacházet elegantně.

Následující kód používá operátor `instanceof` a ukazuje, jak extrahovat data z těla textové zprávy:

```
if (inMessage instanceof TextMessage) {
  String replyString = ((TextMessage) inMessage).getText();
  .
  .
  .
} else {
  // Print error message if Message was not a TextMessage.
  System.out.println("Reply message was not a TextMessage");
}
```

Pokud aplikace odešle zprávu v rámci transakce, zpráva nebude doručena do místa určení, dokud nebude transakce potvrzena. To znamená, že aplikace nemůže odeslat zprávu a přijmout odpověď na zprávu v rámci stejné transakce.

Pokud spotřebitel zpráv přijímá zprávy z místa určení, které je konfigurováno pro dopředné čtení, všechny přechodné zprávy, které jsou ve vyrovnávací paměti čtení napřed při ukončení aplikace, budou vyřazeny.

V doméně publikování/odběru produkt JMS identifikuje dva typy odběratelů zpráv, přechodného odběratele tématu a trvalého odběratele tématu, které jsou popsány v následujících dvou sekcích.

Odběratelé tématu Nondurable

Odběratel netrvalého tématu přijímá pouze zprávy, které byly publikovány v době, kdy je odběratel aktivní. Netrvalý odběr se spustí, když aplikace vytvoří netrvalého odběratele tématu a končí, když aplikace zavře odběratele, nebo když odběratel přestane mít rozsah. Jako rozšíření produktu IBM MQ classes for JMStaké odběratel netrvalého tématu přijímá i zachované publikace.

Chcete-li vytvořit netrvalého odběratele tématu, může aplikace použít metodu createConsumer() nezávislou na doméně a určit jako cíl objekt Topic. Alternativně může aplikace použít metodu specifickou pro doménu createSubscriber(), jak je uvedeno v následujícím příkladu:

```
TopicSubscriber subscriber = session.createSubscriber(topic);
```

Parametr `topic` je objekt Topic, který aplikace vytvořila dříve.

Odběratelé trvalého tématu

Omezení: Aplikace nemůže vytvořit trvalé odběratele tématu při použití připojení v reálném čase ke zprostředkovateli.

Odběratel trvalého tématu přijímá všechny zprávy, které byly publikovány během životnosti trvalého odběru. Tyto zprávy zahrnují všechny ty zprávy, které jsou publikovány v době, kdy odběratel není aktivní. Jako rozšíření v produktu IBM MQ classes for JMSzískává odběratel trvalých témat také zachované publikace.

Chcete-li vytvořit odběratele trvalého tématu, aplikace použije metodu createDurableSubscriber () objektu Session, jak ukazuje následující příklad:

```
TopicSubscriber subscriber = session.createDurableSubscriber(topic, "D_SUB_000001");
```

Na volání createDurableSubscriber () je první parametr objekt Topic, který aplikace vytvořila dříve, a druhý parametr je název, který se používá k identifikaci trvalého odběru.

Relace použitá k vytvoření odběratele trvalého tématu musí mít přidružený identifikátor klienta. Identifikátor klienta přidružený k relaci je stejný jako identifikátor klienta pro připojení, které se používá k vytvoření relace. Identifikátor klienta lze zadat tak, že nastavíte vlastnost CLIENTID objektu ConnectionFactory . Alternativně může aplikace uvést identifikátor klienta voláním metody ID () ID objektu setClientobjektu Connection.

Název použitý k identifikaci trvalého odběru musí být jedinečný pouze v rámci identifikátoru klienta, a proto je identifikátor klienta součástí úplného, jedinečného identifikátoru trvalého odběru. Chcete-li nadále používat trvalý odběr, který byl vytvořen dříve, musí aplikace vytvořit trvalého odběratele témat pomocí relace se stejným identifikátorem klienta, jaký je přidružen k trvalému odběru, a s použitím stejného názvu odběru.

Trvalý odběr se spustí, když aplikace vytvoří trvalého odběratele tématu pomocí identifikátoru klienta a názvu odběru, pro který momentálně neexistuje trvalý odběr. Trvalý odběr však není ukončen, pokud aplikace zavře odběratele trvalého tématu. Chcete-li ukončit trvalý odběr, musí aplikace volat metodu unsubscribe () objektu relace, který má stejný identifikátor klienta jako přidružený k trvalému odběru. Parametr ve volání zrušení odběru () je názvem odběru, jak je zobrazeno v následujícím příkladu:

```
session.unsubscribe("D_SUB_000001");
```

Rozsah trvání trvalého odběru je správce front. Pokud existuje trvalý odběr v jednom správci front a aplikace připojená k jinému správci front vytvoří trvalý odběr se stejným identifikátorem klienta a názvem odběru, jsou dva trvalé odběry zcela nezávislé.

Voliče zpráv

Aplikace může určit, že po sobě jdoucí volání receive () vrátí pouze zprávy, které splňují určitá kritéria. Při vytváření objektu MessageConsumer může aplikace určit výraz jazyka SQL (Structured Query Language), který určuje, které zprávy jsou načítány. Tento výraz SQL se nazývá *selektor zpráv*. Selektor zpráv může obsahovat názvy polí záhlaví zpráv produktu JMS a vlastností zprávy. Informace o tom, jak vytvořit selektor zpráv, viz “Selektory zpráv v JMS” na stránce 122.

Následující příklad ukazuje, jak aplikace může vybírat zprávy založené na vlastnosti definované uživatelem s názvem myProp:

```
MessageConsumer consumer;
.
consumer = session.createConsumer(destination, "myProp = 'blue'");
```

Specifikace JMS nepovoluje aplikaci změnit selektor zpráv pro spotřebitele zpráv. Poté, co aplikace vytvoří spotřebitele zpráv se selektorem zpráv, zůstane selektor zpráv po celou dobu životnosti tohoto spotřebitele. Pokud aplikace vyžaduje více než jeden selektor zpráv, aplikace musí vytvořit spotřebitele zpráv pro každý selektor zpráv.

Všimněte si, že je-li aplikace připojena ke správci front produktu IBM WebSphere MQ 7 , vlastnost MSGSELECTION továrny připojení nemá žádný efekt. Chcete-li optimalizovat výkon, všechny volby zpráv provádí správce front.

Potlačení lokálních publikování

Aplikace může vytvořit spotřebitele zpráv, který ignoruje publikování publikované ve vlastním připojení spotřebitele. Aplikace to provede nastavením třetího parametru na volání createConsumer() na `true`, jak je uvedeno v následujícím příkladu:

```
MessageConsumer consumer = session.createConsumer(topic, null, true);
```

Na volání createDurableSubscriber () to aplikace provede nastavením čtvrtého parametru na hodnotu `true`, jak je uvedeno v následujícím příkladu.

```
String selector = "company = 'IBM'";
TopicSubscriber subscriber = session.createDurableSubscriber(topic, "D_SUB_000001",
                                                              selector, true);
```

Asynchronní doručení zpráv

Aplikace může přijímat zprávy asynchronně pomocí registrace modulu listener zpráv se spotřebitelem zpráv. Modul listener pro zprávy má metodu s názvem onMessage, která se volá asynchronně, je-li k dispozici vhodná zpráva a jejímž účelem je zpracování zprávy. Následující kód ilustruje tento mechanizmus:

```
import javax.jms.*;

public class MyClass implements MessageListener
{
  // The method that is called asynchronously when a suitable message is available
  public void onMessage(Message message)
  {
    System.out.println("Message is "+message);

    // The code to process the message
    .
    .
    .
  }
}
.
.
.
// Main program (possibly in another class)
.
// Creating the message listener
MyClass listener = new MyClass();

// Registering the message listener with a message consumer
consumer.setMessageListener(listener);

// The main program now continues with other processing
```

Aplikace může použít relaci buď pro příjem zpráv synchronně pomocí volání receive (), nebo pro asynchronní příjem zpráv pomocí listenerů zpráv, ale ne pro obojí. Musí-li aplikace přijímat zprávy synchronně a asynchronně, musí vytvořit samostatné relace.

Jakmile je relace nastavena k asynchronnímu příjmu zpráv, nemohou být na této relaci nebo na objektech vytvořených z této relace volány následující metody:

- MessageConsumer.receive ()
- MessageConsumer.receive (long)
- MessageConsumer.receiveNoWait ()
- Session.acknowledge()
- MessageProducer.send (Místo určení, Zpráva)
- MessageProducer.send (cíl, zpráva, int, int, long)
- MessageProducer.send (Message)
- MessageProducer.send (Message, int, int, long).
- MessageProducer.send (cíl, zpráva, CompletionListener)
- MessageProducer.send (Místo určení, Zpráva, int, int, long, CompletionListener)
- MessageProducer.send (Message, CompletionListener)
- MessageProducer.send (Message, int, int, long, CompletionListener)
- Session.commit()
- Session.createBrowser(Fronta)
- Session.createBrowser(Fronta, Řetězec)
- Session.createBytesMessage()
- Session.createConsumer(Cíl)
- Session.createConsumer(Cíl, String, logická hodnota)
- Session.createDurableSubscriber(Topic, String)
- Session.createDurableSubscriber(Topic, String, String, boolean)
- Session.createMapMessage()
- Session.createMessage()
- Session.createObjectMessage()
- Session.createObjectMessage(serializovatelná)
- Session.createProducer(Cíl)
- Session.createQueue(String)
- Session.createStreamMessage()
- Session.createTemporaryQueue()
- Session.createTemporaryTopic()
- Session.createTextMessage()
- Session.createTextMessage(Řetězec)
- Session.createTopic()
- Session.getAcknowledgeMode()
- Session.getMessageListener()
- Session.getTransacted()
- Session.rollback()
- Session.unsubscribe(String)

Je-li vyvolána některá z těchto metod, bude vyvolána výjimka JMSException obsahující zprávu:

```
JMSCC0033: Volání synchronní metody není povoleno, je-li asynchronní relace používána
asynchronně: 'název metody'
```

je vyhozena.

Příjem nezpracovatelných zpráv

Aplikace může přijmout zprávu, která nemůže být zpracována. Může existovat několik důvodů, proč zprávu nelze zpracovat, například zpráva může mít chybný formát. Takové zprávy jsou popsány jako nezpracovatelné zprávy a vyžadují speciální zacházení, aby se zabránilo rekurzivnímu zpracování zprávy.

Podrobnosti o zpracování nezpracovatelných zpráv najdete v tématu “Zpracování nezpracovatelných zpráv v produktu IBM MQ classes for JMS” na stránce 203.

V 9.0.0.2 V 9.0.2 *Načtení uživatelských dat odběru*

Pokud zprávy, které aplikace IBM MQ classes for JMS spotřebovávají z fronty, jsou z administrativně definovaného trvalého odběru uvedeny administrativně, musí aplikace přistupovat k informacím o uživateli, které jsou k odběru přidruženy. Tyto informace se přidají do zprávy jako vlastnost.

Je-li v produktu Continuous Deliveryz IBM MQ 9.0.2zpráva spotřebována z fronty, která obsahuje záhlaví RFH2 se složkou MQPS, je hodnota přidružená ke klíči Sud, pokud existuje, přidána jako vlastnost řetězce do objektu zprávy JMS vráceného do aplikace IBM MQ classes for JMS . Chcete-li povolit načtení této vlastnosti ze zprávy, lze použít konstantu JMS_IBM_SUBSCRIPTION_USER_DATA v rozhraní JmsConstants s metodou javax.jms.Message.getStringProperty(java.lang.String) k získání uživatelských dat odběru.

V produktu IBM MQ 9.0.0 Fix Pack 2 je v produktu Long Term Supportk dispozici také konstanta JMS_IBM_SUBSCRIPTION_USER_DATA.

V následujícím příkladu je definován administrativní trvalý odběr s použitím příkazu MQSC **DEFINE SUB**:

```
DEFINE SUB('MY.SUBCRIPTION') TOPICSTR('PUBLIC') DEST('MY.SUBSCRIPTION.Q')
USERDATA('Administrative durable subscription to put message to the queue MY.SUBSCRIPTION.Q')
```

Kopie zpráv, které jsou publikovány do řetězce tématu PUBLIC , jsou vloženy do fronty MY.SUBSCRIPTION.Q. Uživatelská data, která jsou přidružená k trvalému odběru, jsou poté přidána jako vlastnost do zprávy, která je uložena ve složce MQPS záhlaví RFH2 s klíčem Sud.

Aplikace IBM MQ classes for JMS může volat:

```
javax.jms.Message.getStringProperty(JmsConstants.JMS_IBM_SUBSCRIPTION_USER_DATA);
```

Pak se vrátí následující řetězec:

```
Administrative durable subscription to put message to the queue MY.SUBSCRIPTION.Q
```

.

Související pojmy

“Záhlaví MQRFH2 a JMS” na stránce 127

Související informace

Definování administrativního odběru

DEFINE SUB

Rozhraní JmsConstants

Zavření aplikace IBM MQ classes for JMS

Je důležité, aby aplikace IBM MQ classes for JMS před zastavením explicitně zavřela určité objekty JMS . Finalizéry nemusí být volány, takže se na ně nespoléhejte na volné prostředky. Nepovolujte ukončení aplikace s aktivním trasovaným trasováním.

Samotné uvolňování paměti nemůže včas uvolnit všechny prostředky IBM MQ classes for JMS a IBM MQ , zvláště pokud aplikace vytváří mnoho objektů s krátkou životností JMS na úrovni relace nebo nižší. Je proto důležité, aby aplikace zavřela objekt Connection, Session, MessageConsumernebo objekt MessageProducer , když již není potřeba.

Pokud aplikace skončí bez zavření připojení, dojde k implicitnímu odvolání transakce pro všechny relace obsažené v relaci s transakcemi. Chcete-li zajistit, aby byly provedeny změny provedené aplikací, před zavřením aplikace uzavřete spojení explicitně.

V aplikaci nepoužívejte finalizéry k zavření objektů produktu JMS . Protože nelze moduly finalizer volat, prostředky nemusí být uvolněny. Když je spojení uzavřeno, zavře všechny relace, které z ní byly vytvořeny. Podobně platí, že MessageConsumers a MessageProducers vytvořené z relace jsou zavřeny při zavření relace. Nicméně, zvažte uzavření relací, MessageConsumersa MessageProducers explicitně k zajištění toho, že prostředky budou uvolněny včas.

Je-li aktivována komprese trasování, System.Halt() vypnutí a abnormální, nekontrolované ukončení prostředí JVM pravděpodobně povede k poškození trasovacího souboru. Je-li to možné, vypněte trasovací prostředek, pokud jste shromáždili trasovací informace, které potřebujete. Pokud sledujete aplikaci až do abnormálního ukončení, použijte nekomprimovaný trasovací výstup.

Poznámka: Chcete-li se odpojit od správce front, vyvolá aplikace platformy JMS metodu close () na objektu připojení.

Zpracování nezpracovatelných zpráv v produktu IBM MQ classes for JMS

Je nezpracovatelná zpráva, kterou nelze zpracovat přijímající aplikací. Pokud je nezpracovatelná zpráva doručena aplikaci a odvolána, může ji produkt IBM MQ classes for JMS přesunout do fronty vyřazených zpráv.

nezpracovatelná zpráva je zpráva, kterou nelze zpracovat přijímající aplikací. Zpráva může mít neočekávaný typ nebo obsahovat informace, které nelze zpracovat logikou aplikace. Je-li do aplikace doručena nezpracovatelná zpráva, aplikace ji nebude schopna zpracovat a vrátí ji zpět do fronty, ze které pochází. Ve výchozím nastavení IBM MQ classes for JMS opakovaně doručí zprávu aplikaci. To může vést k tomu, že aplikace se trvale zasekne ve smyčce a pokusí se zpracovat nezpracovatelnou zprávu a vrátit ji zpět.

Aby k tomu nedocházelo, může produkt IBM MQ classes for JMS zjišťovat nezpracovatelné zprávy a přesunout je do alternativního cíle. Za tímto účelem produkt IBM MQ classes for JMS využívá následující vlastnosti:

- Hodnota pole BackoutCount v rámci MQMD zprávy, která byla zjištěna.
- Atributy fronty IBM MQ **BOTHRESH** (prahová hodnota vrácení) a **BOQNAME** (fronta vrácených zpráv fronty) pro vstupní frontu obsahující zprávu.

Kdykoli je zpráva odvolána aplikací, správce front automaticky zvýší hodnotu pole BackoutCount pro danou zprávu.

Když IBM MQ classes for JMS zjistí zprávu, která má BackoutCount větší než nula, porovnávají hodnotu BackoutCount na hodnotu atributu **BOTHRESH** .

- Pokud je hodnota BackoutCount menší než hodnota atributu **BOTHRESH** , IBM MQ classes for JMS ji doručí do aplikace ke zpracování.
- Pokud je však hodnota BackoutCount větší nebo rovna než **BOTHRESH**, pak se zpráva považuje za nezpracovatelnou zprávu. V této situaci pak IBM MQ classes for JMS přesune zprávu do fronty zadané atributem **BOQNAME** . Pokud zpráva nemůže být vložena do fronty vyřazených zpráv, přesune se do fronty nedoručených zpráv správce front nebo bude vyřazena v závislosti na volbách sestavy ve zprávě.

Poznámka:

- Pokud je atribut **BOTHRESH** ponechán na své výchozí hodnotě 0, zpracování nezpracovatelných zpráv je vypnuto. To znamená, že všechny nezpracovatelné zprávy jsou vraty zpět do vstupní fronty.
- Druhá věc, kterou je třeba poznamenat, je, že IBM MQ classes for JMS dotaz na atributy **BOTHRESH** a **BOQNAME** pro frontu při prvním zjištění zprávy, která má BackoutCount větší než nula. Hodnoty těchto atributů se poté ukládají do mezipaměti a používají se vždy, když se v produktu IBM MQ classes for JMS narazí na zprávu s hodnotou BackoutCount větší než nula.

Konfigurace systému pro zpracování nezpracovatelných zpráv

Fronta, kterou produkt IBM MQ classes for JMS používá při zjišťování atributů **BOTHRESH** a **BOQNAME** , závisí na stylu prováděného systému zpráv:

- Pro systém zpráv typu point-to-point je to základní lokální fronta. To je důležité, pokud aplikace JMS spotřebovává zprávy buď z front aliasů, nebo z front klastru.
- Pro systém zpráv typu publikování/odběr se vytvoří spravovaná fronta, která bude uchovávat zprávy pro aplikaci. IBM MQ classes for JMS dotaz na spravovanou frontu za účelem určení hodnot pro atributy **BOTHRESH** a **BOQNAME** .

  Spravovaná fronta je vytvořena z modelové fronty přidružené k objektu Topic, k jehož odběru se aplikace přihlásal, a dědí hodnoty atributů **BOTHRESH** a **BOQNAME** z modelové fronty. Použitá modelová fronta závisí na tom, zda přijímající aplikace odebrala trvalý nebo netrvalý odběr:

  – Modelová fronta použitá pro trvalé odběry je určena atributem **MDURMDL** objektu Topic. Výchozí hodnota tohoto atributu je `SYSTEM.DURABLE.MODEL.QUEUE`.
  – U netrvalých odběrů je použitá modelová fronta určena atributem **MNDURMDL** . Výchozí hodnota atributu **MNDURMDL** je `SYSTEM.NDURABLE.MODEL.QUEUE`.

Při zjišťování atributů **BOTHRESH** a **BOQNAME** se používá IBM MQ classes for JMS:

- Otevřete lokální frontu nebo cílovou frontu pro alias frontu.
- Dotažte se na atributy **BOTHRESH** a **BOQNAME** .
- Zavřete lokální frontu nebo cílovou frontu pro alias frontu.

Volby otevření, které se používají při otevírání lokální fronty nebo cílové fronty pro alias frontu, závisí na verzi používaného produktu IBM MQ classes for JMS :

- Pokud při použití IBM MQ classes for JMS pro IBM MQ 9.0.0 Fix Pack 5 a starší, je-li lokální fronta nebo cílová fronta pro alias frontu, je fronta klastru, pak IBM MQ classes for JMS otevřete frontu s volbami `MQOO_INPUT_AS_Q_DEF`, `MQOO_INQUIRE` a `MQOO_FAIL_IF_QUIESCING` . To znamená, že uživatel, který spouští přijímající aplikaci, musí mít dotazům a získat přístup k lokální instanci fronty klastru.

  IBM MQ classes for JMS otevře všechny ostatní typy lokální fronty s volbami otevření `MQOO_INQUIRE` a `MQOO_FAIL_IF_QUIESCING`. Aby mohl produkt IBM MQ classes for JMS zadávat dotazy na hodnoty atributů, musí mít uživatel, který spouští přijímající aplikaci, přístup k dotazům na lokální frontě.
- V 9.0.0.6 Při použití produktu IBM MQ classes for JMS pro produkt IBM MQ 9.0.0 Fix Pack 6 a pozdější musí mít uživatel, který spouští přijímající aplikaci, dotazům přístup k lokální frontě, bez ohledu na typ fronty.

Chcete-li přesunout nezpracovatelné zprávy do fronty vrácených zpráv do fronty nebo do fronty nedoručených zpráv správce front, musíte uživateli udělit oprávnění `put` a `passall` oprávnění aplikace.

Zpracování nezpracovatelných zpráv pro synchronní aplikace

Pokud aplikace přijímá zprávy synchronně, vyvoláním jedné z následujících metod IBM MQ classes for JMS znovu zařadit nezpracovatelnou zprávu do jednotky práce, která byla aktivní, když se aplikace pokusila získat zprávu:

- JMSConsumer.receive()
- JMSConsumer.receive(dlouhý časový limit)
- JMSConsumer.receiveBody(Třída < T> c)
- JMSConsumer.receiveBody(Třída < T> c, dlouhý časový limit)
- JMSConsumer.receiveBodyNoWait Třída < T> c)
- JMSConsumer.receiveNoWait()
- MessageConsumer.receive ()
- MessageConsumer.receive (long timeout)

- MessageConsumer.receiveNoWait ()
- QueueReceiver.receive ()
- QueueReceiver.receive (dlouhý časový limit)
- QueueReceiver.receiveNoWait ()
- TopicSubscriber.receive ()
- TopicSubscriber.receive (long timeout)
- TopicSubscriber.receiveNoWait ()

This means that if the application is using either a transacted JMS context or session, then the moving of the message to the backout queue is not committed until the transaction is committed.

Je-li atribut **BOTHRESH** nastaven na jinou hodnotu než nula, měl by být nastaven i atribut **BOQNAME** . Je-li hodnota **BOTHRESH** nastavena na hodnotu větší než nula a **BOQNAME** nebyla nastavena, chování je určeno volbami sestavy ve zprávě:

- Má-li zpráva volbu sestavy MQRO_DISCARD_MSG , zpráva se zahodí.
- Má-li zpráva určenou volbu sestavy MQRO_DEAD_LETTER_Q , pokusí se příkaz IBM MQ classes for JMS přesunout zprávu do fronty nedoručených zpráv správce front.
- Pokud zpráva nemá nastavenou hodnotu MQRO_DISCARD_MSG nebo MQRO_DEAD_LETTER_Q , pokusí se příkaz IBM MQ classes for JMS o vložení zprávy do fronty nedoručených zpráv pro správce front.

V případě, že se pokus o vložení zprávy do fronty nedoručených zpráv z nějakého důvodu nezdaří, bude to, co se stane s touto zprávou, určeno, zda přijímající aplikace používá kontext nebo relaci, která neobsahuje transakci JMS nebo zda obsahuje relaci:

- Pokud přijímající aplikace používá buď kontext nebo relaci s transakčním JMS , a transakce je potvrzena, pak bude zpráva vyřazena.
- Pokud přijímající aplikace používá kontext nebo relaci transakce JMS a transakci odvolá zpět, vrátí se zpráva do vstupní fronty.
- Pokud přijímající aplikace vytvořila kontext nebo relaci produktu JMS bez transakce, bude zpráva vyřazena.

Zpracování nezpracovatelných zpráv pro asynchronní aplikace

Pokud aplikace přijímá zprávy asynchronně prostřednictvím MessageListener, IBM MQ classes for JMS znovu nezpracovatelné zprávy bez ovlivnění doručení zprávy. Proces opětovného zařazení je k dispozici mimo jakoukoli jednotku práce přidruženou ke skutečnému doručení zprávy do aplikace.

Je-li hodnota **BOTHRESH** nastavena na hodnotu větší než nula a **BOQNAME** nebyla nastavena, chování je určeno volbami sestavy ve zprávě:

- Má-li zpráva volbu sestavy MQRO_DISCARD_MSG , zpráva se zahodí.
- Má-li zpráva určenou volbu sestavy MQRO_DEAD_LETTER_Q , pokusí se příkaz IBM MQ classes for JMS přesunout zprávu do fronty nedoručených zpráv správce front.
- Pokud zpráva nemá nastavenou hodnotu MQRO_DISCARD_MSG nebo MQRO_DEAD_LETTER_Q , pokusí se příkaz IBM MQ classes for JMS o vložení zprávy do fronty nedoručených zpráv pro správce front.

Pokud se pokus o vložení zprávy do fronty nedoručených zpráv z nějakého důvodu nezdaří, IBM MQ classes for JMS vrátí zprávu do vstupní fronty.

Informace o tom, jak specifikace aktivace a ConnectionConsumers zpracovávají nezpracovatelné zprávy, najdete v tématu Odebrání zpráv z fronty v aplikaci ASF.

Co se stane se zprávou, když je přesunuta do fronty vyřazovacích zpráv

When a poison message is requeued to the backout requeue queue, the IBM MQ classes for JMS add an RFH2 header to it (if it did not have one already), and update some of the fields within the message descriptor (MQMD).

Obsahuje-li nezpracovatelná zpráva záhlaví RFH2 (protože se jednalo například o zprávu JMS), změní IBM MQ classes for JMS následující pole v rámci MQMD při přesunu zprávy do fronty vrácených zpráv:

- Pole BackoutCount je resetováno na nulu.
- Pole Vypršení platnosti zprávy je aktualizováno tak, aby odráželo zbývající vypršení platnosti v době přijetí nezpracovatelné zprávy aplikací produktu JMS .

Pokud nezpracovatelná zpráva neobsahuje záhlaví RFH2 , přidá IBM MQ classes for JMS jedno a aktualizuje následující pole v deskriptoru MQMD jako součást zpracování odvolání:

- Pole BackoutCount je resetováno na nulu.
- Pole Vypršení platnosti zprávy je aktualizováno tak, aby odráželo zbývající vypršení platnosti v době přijetí nezpracovatelné zprávy aplikací produktu JMS .
- Pole Formát zprávy se změní na MQHRF2.
- Pole CCSID se změní na 1208.
- Pole Kódování je upraveno na 273.

Kromě toho se pole CCSID a kódování z nezpracovatelné zprávy kopírují do polí CCSID a kódování záhlaví RFH2 , aby bylo zajištěno, že řetězení záhlaví zprávy ve frontě vrácených zpráv je správné.

Související pojmy

“Zpracování nezpracovatelných zpráv v ASF” na stránce 298
V rámci služeb aplikačního serveru je zpracování nezpracovatelných zpráv ošetřeno nepatrně jiným způsobem v produktu IBM MQ classes for JMS.

Výjimky v IBM MQ classes for JMS

Aplikace produktu IBM MQ classes for JMS musí být schopna zpracovávat výjimky vyvolané voláním rozhraní API produktu JMS nebo doručené do obslužné rutiny výjimek.

Produkt IBM MQ classes for JMS vykazuje běhové problémy vyvoláním výjimek. JMSException je kořenová třída pro výjimky vyvolané metodami JMS a zachycení výjimek JMSException poskytuje generický způsob obsluhy všech souvisejících výjimek produktu JMS .

Každá výjimka JMSException zapouzdřuje následující informace:

- Zpráva o výjimce specifické pro poskytovatele, kterou aplikace získá voláním metody Throwable.getMessage().
- Kód chyby specifický pro poskytovatele, který aplikace získá voláním metody JMSException.getErrorCode().
- Propojená výjimka. Výjimka vyvolaná voláním rozhraní API produktu JMS je často výsledkem problému nižší úrovně, který je hlášen jinou výjimkou spojenou s touto výjimkou. Aplikace získá spojenou výjimku voláním metody JMSException.getLinkedException() nebo metody Throwable.getCause().

Většina výjimek vygenerovaných produktem IBM MQ classes for JMS jsou instance podtříd výjimky JMSException. Tyto podtřídy implementují rozhraní com.ibm.msg.client.jms.JmsExceptionDetail , které poskytuje následující dodatečné informace:

- Vysvětlení zprávy o výjimce, kterou aplikace získá voláním metody JmsExceptionDetail.getExplanation().
- Doporučená odezva uživatele na výjimku, kterou aplikace získá voláním metody JmsExceptionDetail.getUserAction().
- Klíče pro vložení zpráv ve zprávě výjimky. Aplikace získá iterátor pro všechny klíče pomocí volání metody JmsExceptionDetail.getKeys().
- Zpráva se vloží do zprávy výjimky. Vložením zprávy může být například název fronty, která výjimku způsobila, a může být užitečné, aby aplikace mohla přistupovat k tomuto názvu. Aplikace získá vložení zprávy odpovídající uvedenému klíči voláním metody JmsExceptionDetail.getValue().

Všechny metody v rozhraní Podrobnosti JmsExceptionmohou vracet hodnotu null, nejsou-li k dispozici žádné podrobnosti.

Pokud se například aplikace pokusí vytvořit producenta zpráv pro frontu IBM MQ , která neexistuje, je vyvolána výjimka s následujícími informacemi:

```
Message : JMSWMQ2008: Failed to open MQ queue 'Q_test'.
Class : class com.ibm.msg.client.jms.DetailedInvalidDestinationException
Error Code : JMSWMQ2008
Explanation : JMS attempted to perform an MQOPEN, but IBM MQ reported an
              error.
User Action : Use the linked exception to determine the cause of this error. Check
              that the specified queue and queue manager are defined correctly.
```

Vyvolaná výjimka com.ibm.msg.client.jms.DetailedInvalidDestinationExceptionje podtřídou třídy javax.jms.InvalidDestinationException a implementuje rozhraní com.ibm.msg.client.jms.JmsExceptionDetail .

Propojené výjimky

Propojená výjimka poskytuje další informace o běhovém problému. Proto pro každou výjimku JMSException, která je vyvolána, by měla aplikace zkontrolovat připojenou výjimku. Samotná připojená výjimka může mít jinou propojené výjimky a propojené výjimky tvoří řetězec vedoucí zpět k původnímu základnímu problému. Propojená výjimka je implementována použitím mechanizmu výjimek v řetězové výjimce třídy java.lang.Throwable a aplikace obdrží připojenou výjimku voláním metody Throwable.getCause(). Pro výjimku JMSException metoda getLinkedException () ve skutečnosti deleguje metodu Throwable.getCause().

Pokud například aplikace určuje nesprávné číslo portu při připojování ke správci front, budou tyto výjimky tvořit následující řetězec:

```
com.ibm.msg.client.jms.DetailIllegalStateException
|
+- -->
com.ibm.mq.MQException
|
+- -->
com.ibm.mq.jmqi.JmqiException
|
+- -->
java.net.ConnectionException
```

Obvykle je každá výjimka v řetězci vyvolána z jiné vrstvy v kódu. Například výjimky v předchozím řetězci jsou vyvoláženy následujícími vrstvami:

- První výjimka, instance podtřídy JMSException, je vyvolána společnou vrstvou v produktu IBM MQ classes for JMS.
- Další výjimka, instance com.ibm.mq.MQException, je vyvolána poskytovatelem systému zpráv produktu IBM MQ .
- Další výjimka: instance com.ibm.mq.jmqi.JmqiException, je vyvolána společným rozhraním produktu Java pro rozhraní MQI.
- Poslední výjimka, instance java.net.ConnectionException, je vyvolána knihovnou tříd Java .

Další informace o vrstvené architektuře produktu IBM MQ classes for JMSnaleznete v tématu TřídyIBM MQ pro architekturu JMS.

Pomocí kódu podobného následujícímu kódu může aplikace iterovat přes tento řetězec a extrahovat všechny příslušné informace:

```
import com.ibm.msg.client.jms.JmsExceptionDetail;
import com.ibm.mq.MQException;
import com.ibm.mq.jmqi.JmqiException;
import javax.jms.JMSException;
.
.
.
catch (JMSException je) {
System.err.println("Caught JMSException");
```

```
// Check for linked exceptions in JMSException
Throwable t = je;
while (t != null) {
// Write out the message that is applicable to all exceptions
System.err.println("Exception Msg: " + t.getMessage());
// Write out the exception stack trace
t.printStackTrace(System.err);

// Add on specific information depending on the type of exception
if (t instanceof JMSException) {
JMSException je1 = (JMSException) t;
System.err.println("JMS Error code: " + je1.getErrorCode());

if (t instanceof JmsExceptionDetail){
JmsExceptionDetail jed = (JmsExceptionDetail)je1;
System.err.println("JMS Explanation: " + jed.getExplanation());
System.err.println("JMS Explanation: " + jed.getUserAction());
}
} else if (t instanceof MQException) {
MQException mqe = (MQException) t;
System.err.println("WMQ Completion code: " + mqe.getCompCode());
System.err.println("WMQ Reason code: " + mqe.getReason());
} else if (t instanceof JmqiException){
JmqiException jmqie = (JmqiException)t;
System.err.println("WMQ Log Message: " + jmqie.getWmqLogMessage());
System.err.println("WMQ Explanation: " + jmqie.getWmqMsgExplanation());
System.err.println("WMQ Msg Summary: " + jmqie.getWmqMsgSummary());
System.err.println("WMQ Msg User Response: "
+ jmqie.getWmqMsgUserResponse());
System.err.println("WMQ Msg Severity: " + jmqie.getWmqMsgSeverity());
}

// Get the next cause
t = t.getCause();
}
}
```

Vezměte na vědomí, že aplikace by měla vždy kontrolovat typ každé výjimky v řetězu, protože typ výjimky se může lišit a výjimky různých typů zapouzdřují různé informace.

Získání specifických informací IBM MQ o problému

Instance com.ibm.mq.MQException a com.ibm.mq.jmqi.JmqiException zapouzdřují IBM MQ specifické informace o problému.

Výjimka MQException zapouzdřuje následující informace:

- Kód dokončení, který aplikace získá voláním metody Code () getComp
- Kód příčiny, který aplikace získá voláním metody getReason()

Výjimka JmqiException také zapouzdřuje kód dokončení a kód příčiny. Kromě toho však výjimka JmqiException zapouzdřuje informace ve zprávě AMQ *nnnn* nebo CSQ *nnnn* , je-li k výjimce přidružena výjimka. Voláním odpovídajících metod této výjimky může aplikace získat různé komponenty této zprávy, jako je závažnost, vysvětlení a odezva uživatele.

Příklady použití metod uvedených v tomto oddílu naleznete v ukázkovém kódu v produktu “Propojené výjimky” na stránce 207.

Upgrade z předchozích verzí produktu IBM MQ classes for JMS

V porovnání s předchozími verzemi produktu IBM MQ classes for JMSse většina chybových kódů a zpráv výjimek změnila v IBM WebSphere MQ 7.0. Důvodem těchto změn je to, že z produktu IBM WebSphere MQ 7.0má produkt IBM MQ classes for JMS vrstvenou architekturu a výjimky jsou vyvoláženy z různých vrstev kódu.

Pokud se například aplikace pokouší o připojení ke správci front, který neexistuje, předchozí verze produktu IBM MQ classes for JMS vygenerovala výjimku JMSException s následujícími informacemi:

```
MQJMS2005: Failed to create MQQueueManager for 'localhost:QM_test'.
```

Tato výjimka obsahovala výjimku propojené výjimky MQException s následujícími informacemi:

```
MQJE001: Completion Code 2, Reason 2058
```

Ve srovnání se stejnými okolnostmi příkaz IBM MQ classes for JMS v produktu IBM WebSphere MQ 7.0 vyvolá výjimku JMSException s následujícími informacemi:

```
Message : JMSWMQ0018: Failed to connect to queue manager 'QM_test' with
connection mode 'Client' and host name 'localhost'.
Class : class com.ibm.msg.client.jms.DetailedJMSException
Error Code : JMSWMQ0018
Explanation : null
User Action : Check the queue manager is started and if running in client mode,
check there is a listener running. Please see the linked exception
for more information.
```

Tato výjimka obsahuje spojenou výjimku MQException s následujícími informacemi:

```
Message : JMSCMQ0001: IBM MQ call failed with compcode '2' ('MQCC_FAILED')
          reason '2058' ('MQRC_Q_MGR_NAME_ERROR').
Class : class com.ibm.mq.MQException
Completion Code : 2
Reason Code : 2058
```

Pokud vaše aplikace analyzuje nebo testuje zprávy o výjimce vrácené metodou Throwable.getMessage() nebo kódy chyb vrácené metodou JMSException.getErrorCode() a provádíte upgrade z vydání před produktem IBM WebSphere MQ 7.0, bude pravděpodobně třeba upravit aplikaci tak, aby používala produkt IBM MQ classes for JMS v produktu IBM WebSphere MQ 7.0.

Moduly listener výjimek

Aplikace může registrovat modul listener pro výjimky s objektem připojení. Pokud se následně vyskytne problém, který znemožňuje připojení, produkt IBM MQ classes for JMS doručí výjimku do modulu listener výjimek vyvoláním metody onException(). Aplikace pak má možnost znovu ustanovit spojení. Produkt IBM MQ classes for JMS může při pokusu o doručení zprávy asynchronně doručit výjimku do modulu listener výjimek.

Chcete-li v produktu IBM MQ 8.0.0 Fix Pack 2zachovat chování pro aktuální aplikace JMS, které konfigurují JMS MessageListener a JMS ExceptionListener, a aby se zajistilo, že je IBM MQ classes for JMS konzistentní se specifikací JMS , výchozí hodnota pro vlastnost ASYNC_EXCEPTIONS JMS ConnectionFactory se změní na ASYNC_EXCEPTIONS_CONNECTIONBROKEN pro IBM MQ classes for JMS. V důsledku toho jsou při výchozím nastavení k aplikaci JMS ExceptionListenerdoručeny pouze výjimky odpovídající chybným kódům chyb připojení.

V 9.0.0.1 APAR IT14820, zahrnutý z produktu IBM MQ 9.0.0 Fix Pack 1, aktualizuje produkt IBM MQ classes for JMS tak, aby:

- Modul ExceptionListener registrovaný aplikací je vyvolán pro jakékoli přerušené výjimky připojení bez ohledu na to, zda aplikace používá synchronní nebo asynchronní spotřebitele zpráv.
- Modul ExceptionListener registrovaný aplikací je vyvolán, pokud je poškozen soket TCP/IP používaný relací produktu JMS .
- Nepřerušené přerušené výjimky (například MQRC_GET_INHIBITED), které vznikají při doručení zprávy, jsou doručeny do aplikace ExceptionListener , když aplikace používá asynchronní spotřebitele zpráv a JMS ConnectionFactory použitá aplikací má vlastnost ASYC_EXCEPTIONS nastavenou na hodnotu ASYNC_EXCEPTIONS_ALL.

Poznámka: Modul ExceptionListener je vyvolán pouze jednou pro poškozenou výjimku připojení, a to i v případě, že jsou přerušena dvě připojení TCP/IP (jedna používaná připojením JMS a jedna používaná relací JMS).

U všech ostatních typů problémů je vyvolána výjimka JMSException aktuální voláním rozhraní API produktu JMS .

Pokud aplikace neregistruje modul listener pro výjimky s objektem připojení, všechny výjimky, které by byly doručeny do modulu listener pro výjimky, jsou zapsány do protokolu IBM MQ classes for JMS for JMS .

Související informace

Architektura produktu IBM MQ pro architekturu JMS

VÝJIMKA ASYNCEXCEPTION

Přístup k funkcím produktu IBM MQ z aplikace IBM MQ classes for JMS

Produkt IBM MQ classes for JMS poskytuje funkce pro využití mnoha funkcí produktu IBM MQ.

Upozornění: Tyto funkce jsou mimo specifikaci JMS , nebo jsou v určitých případech porušeny specifikace JMS . Pokud je použijete, je nepravděpodobné, že by vaše aplikace byla kompatibilní s jinými poskytovateli JMS . Tyto funkce, které nejsou v souladu se specifikací JMS , jsou označeny upozorněním Upozornění.

Čtení a zápis deskriptoru zpráv z aplikace IBM MQ classes for JMS

Možnost přístupu k deskriptoru zpráv (MQMD) lze řídit nastavením vlastností na cíli a ve zprávě.

Některé aplikace produktu IBM MQ vyžadují, aby byly nastaveny specifické hodnoty v deskriptoru MQMD pro zprávy odeslané k nim. Produkt IBM MQ classes for JMS poskytuje atributy zpráv, které umožňují aplikacím produktu JMS nastavit pole MQMD, a umožnit tak aplikacím produktu JMS "řídit" aplikace produktu IBM MQ .

Musíte nastavit vlastnost cílového objektu WMQ_MQMD_WRITE_ENABLED na hodnotu true pro nastavení vlastností MQMD, které mají mít nějaký vliv. Poté můžete použít metody nastavení vlastností pro zprávu (například setStringProperty) k přiřazení hodnot k polím MQMD. Všechna pole MQMD jsou odkryta s výjimkou StrucId a Verze; BackoutCount lze číst, ale ne zapisovat do.

Tento příklad vede k vložení zprávy do fronty nebo tématu s MQMD.UserIdentifier nastaven na "JoeBloggs".

```
// Create a ConnectionFactory, connection, session, producer, message
// ...

// Create a destination
// ...

// Enable MQMD write
dest.setBooleanProperty(WMQConstants.WMQ_MQMD_WRITE_ENABLED, true);

// Optionally, set a message context if applicable for this MD field
dest.setIntProperty(WMQConstants.WMQ_MQMD_MESSAGE_CONTEXT,
  WMQConstants.WMQ_MDCTX_SET_IDENTITY_CONTEXT);

// On the message, set property to provide custom UserId
msg.setStringProperty("JMS_IBM_MQMD_UserIdentifier", "JoeBloggs");

// Send the message
// ...
```

Před nastavením parametru JMS_IBM_MQMD_UserIdentifierje třeba nastavit WMQ_MQMD_MESSAGE_CONTEXT. Další informace o použití WMQ_MQMD_MESSAGE_CONTEXT viz “Vlastnosti objektu zprávy produktu JMS” na stránce 212.

Podobně můžete extrahovat obsah polí MQMD nastavením hodnoty WMQ_MQMD_READ_ENABLED na hodnotu true před přijetím zprávy a následným použitím metod get zprávy, jako je například vlastnost getString. Všechny přijaté vlastnosti jsou jen pro čtení.

Tento příklad má za výsledek pole *hodnota* , které drží hodnotu MQMD.ApplIdentityData pole zprávy bylo získáno z fronty nebo tématu.

```
// Create a ConnectionFactory, connection, session, consumer
```

```
// ...

// Create a destination
// ...

// Enable MQMD read
dest.setBooleanProperty(WMQConstants.WMQ_MQMD_READ_ENABLED, true);

// Receive a message
// ...

// Get MQMD field value using a property
String value = rcvMsg.getStringProperty("JMS_IBM_MQMD_ApplIdentityData");
```

Vlastnosti objektu místa určení JMS
Dvě vlastnosti objektu Cílový objekt řídí přístup k deskriptoru MQMD z produktu JMSa třetí kontext zpráv ovládacích prvků.

Tabulka 35. Názvy a popisy vlastností

Vlastnost	**Krátký formát**	**Popis**
WMQ_MQMD_WRITE_ENABLED	MDW	Určuje, zda může aplikace JMS nastavit hodnoty polí MQMD
WMQ_MQMD_READ_ENABLED	MDR	Zda může aplikace JMS extrahovat hodnoty polí MQMD
KONTEXT WMQ_MQMD_MESSAGE_	MDCTX	Jaká úroveň kontextu zprávy má být nastavena aplikací JMS . Aby tato vlastnost mohla nabýt účinku, musí být aplikace spuštěna s příslušným kontextovým oprávněním.

Tabulka 36. Názvy vlastností, hodnoty a metody nastavení

Vlastnost	**Platné hodnoty v nástroji pro administraci (výchozí nastavení tučně)**	**Platné hodnoty v programech**	**Nastavit metodu**
WMQ_MQMD_WRITE_ENABLED	• **NE** Všechny vlastnosti JMS_IBM_MQMD* jsou ignorovány a jejich hodnoty se nezkopírují do základní struktury MQMD. • YES Vlastnosti JMS_IBM_MQMD* jsou zpracovány. Jejich hodnoty jsou zkopírovány do podkladové struktury MQMD.	• **Nepravda** • Pravda	setMQMDWritePovoleno

Tabulka 36. Názvy vlastností, hodnoty a metody nastavení (pokračování)			
Vlastnost	**Platné hodnoty v nástroji pro administraci (výchozí nastavení tučně)**	**Platné hodnoty v programech**	**Nastavit metodu**
WMQ_MQMD_READ_ENABLED	• **NE** Při odesílání zpráv nejsou vlastnosti JMS_IBM_MQMD* v odeslané zprávě aktualizovány tak, aby odrážely aktualizované hodnoty polí v produktu MQMD. Při příjmu zpráv nejsou dostupné žádné z vlastností JMS_IBM_MQMD* v přijaté zprávě, i když odesilatel některé či všechny tyto vlastnosti nastavil. • YES Při odesílání zpráv jsou všechny vlastnosti JMS_IBM_MQMD* v odeslané zprávě aktualizovány tak, aby odrážely aktualizované hodnoty polí v MQMD, včetně těch, které odesilatel explicitně nenastavila. Při příjmu zpráv jsou dostupné všechny vlastnosti JMS_IBM_MQMD* v přijaté zprávě včetně vlastností, které odesilatel explicitně nenastavil.	• **Nepravda** • Pravda	setMQMDReadPovoleno
WMQ_MQMD_MESSAGE_CONTEXT	• **Výchozí** Volání rozhraní MQOPEN API a struktura MQPMO neurčují žádné explicitní volby kontextu zprávy. • KONTEXT SET_IDENTITY_CONTEXT Volání rozhraní MQOPEN API určuje volbu kontextu zprávy MQOO_SET_IDENTITY_CONTEXT a struktura MQPMO určuje hodnotu MQPMO_SET_IDENTITY_CONTEXT. • NASTAVITEL_VŠECH_KONTEXTU Volání rozhraní MQOPEN API uvádí volbu kontextu zprávy MQOO_SET_ALL_CONTEXT a struktura MQPMO určuje MQPMO_SET_ALL_CONTEXT	• **WMQ_MDCTX_DEFAULT** • KONTEXT WMQ_MDCTX_SET_IDENTITY_CONTEXT • WMQ_MDCTX_SET_ALL_CONTEXT	Kontext setMQMDMessage

Vlastnosti objektu zprávy produktu JMS
Vlastnosti objektu zprávy s předponou JMS_IBM_MQMD umožňují nastavit nebo přečíst odpovídající pole MQMD.

Odesílání zpráv

Všechna pole MQMD s výjimkou StrucId a verze jsou reprezentovány. Tyto vlastnosti se odkazují pouze na pole MQMD, kde se vlastnost vyskytuje jak v deskriptoru MQMD, tak v záhlaví MQRFH2 , verze v MQRFH2 není nastavena ani extrahována.

Je možné nastavit libovolnou z těchto vlastností, kromě JMS_IBM_MQMD_BackoutCount. Jakákoli hodnota nastavená pro parametr JMS_IBM_MQMD_BackoutCount je ignorována.

Má-li vlastnost maximální délku a zadáte příliš dlouhou hodnotu, bude tato hodnota zkrácena.

Pro určité vlastnosti musíte také nastavit vlastnost WMQ_MQMD_MESSAGE_CONTEXT na objektu Destination. Aby tato vlastnost mohla nabýt účinnosti, musí aplikace běžet s příslušným oprávněním kontextu. Pokud nenastavíte hodnotu WMQ_MQMD_MESSAGE_CONTEXT na příslušnou hodnotu, bude hodnota vlastnosti ignorována. Nastavíte-li WMQ_MQMD_MESSAGE_CONTEXT na příslušnou hodnotu, ale nemáte dostatečné oprávnění ke kontextu pro správce front, bude vydána výjimka JMSException. Vlastnosti vyžadující specifické hodnoty WMQ_MQMD_MESSAGE_CONTEXT jsou následující.

Následující vlastnosti vyžadují nastavení WMQ_MQMD_MESSAGE_CONTEXT na hodnotu WMQ_MDCT_X_SET_IDENTITY_CONTEXT nebo WMQ_MDCTX_SET_ALL_CONTEXT:

- JMS_IBM_MQMD_UserIdentifier .
- JMS_IBM_MQMD_AccountingToken .
- JMS_IBM_MQMD_ApplIdentityData

Následující vlastnosti vyžadují nastavení WMQ_MQMD_MESSAGE_CONTEXT na hodnotu WMQ_MDCTX_SET_ALL_CONTEXT:

- JMS_IBM_MQMD_PutApplTyp
- Název JMS_IBM_MQMD_PutAppl
- JMS_IBM_MQMD_PutDate .
- JMS_IBM_MQMD_PutTime .
- JMS_IBM_MQMD_ApplOriginData

Příjem zpráv

Všechny tyto vlastnosti jsou k dispozici v přijaté zprávě, je-li vlastnost WMQ_MQMD_READ_ENABLED nastavena na hodnotu true, bez ohledu na skutečné vlastnosti, které vytvořila aplikace pro vytváření. Aplikace nemůže upravit vlastnosti přijaté zprávy, pokud nejsou nejprve vymazány všechny vlastnosti, podle specifikace JMS . Obdrženou zprávu lze předat bez úpravy vlastností.

Upozornění: Pokud vaše aplikace přijme zprávu z cíle s vlastností WMQ_MQMD_READ_ENABLED nastaveným na hodnotu true a předá ji cíli s hodnotou WMQ_MQMD_WRITE_ENABLED nastavenou na hodnotu true, budou výsledkem všech hodnot pole MQMD přijaté zprávy do předané zprávy.

Tabulka vlastností

V této tabulce jsou uvedeny vlastnosti objektu zpráv reprezentující pole MQMD. Úplné popisy polí a jejich přípustné hodnoty najdete v odkazech.

Tabulka 37. Názvy vlastností, popisy a typy

Vlastnost	Popis	Java Typ	Odkaz na úplný popis
JMS_IBM_MQMD_Report.	Volby pro zprávy sestav	Celé číslo	Sestava
JMS_IBM_MQMD_MsgType	Typ zprávy	Celé číslo	MsgType
JMS_IBM_MQMD_Expiry	Životnost zprávy	Celé číslo	Vypršení
JMS_IBM_MQMD_Feedback.	Zpětná vazba nebo kód příčiny	Celé číslo	Zpětná vazba

Tabulka 37. Názvy vlastností, popisy a typy (pokračování)

Vlastnost	Popis	Java Typ	Odkaz na úplný popis
JMS_IBM_MQMD_Encoding.	Číselný kódování dat zprávy	Celé číslo	Kódování
JMS_IBM_MQMD_CodedCharSetId	Identifikátor znakové sady dat zprávy	Celé číslo	CodedCharSetId
JMS_IBM_MQMD_Format.	Název formátu dat zprávy	Řetězec	Formát
JMS_IBM_MQMD_Priority [1]	Priorita zprávy	Celé číslo	Priorita
JMS_IBM_MQMD_Perzistence	Trvalost zpráv	Celé číslo	Trvání
JMS_IBM_MQMD_MsgId [2]	Identifikátor zprávy	Objekt (byte []) [4]	MsgId
JMS_IBM_MQMD_CorrelId [3]	Identifikátor korelace	Objekt (byte []) [4]	CorrelId
JMS_IBM_MQMD_BackoutCount .	Počítadlo odvolaných	Celé číslo	BackoutCount
JMS_IBM_MQMD_ReplyToQ	Název fronty odpovědí	Řetězec	ReplyToQ
Správce front JMS_IBM_MQMD_ReplyTo	Název správce front odpovědí	Řetězec	ReplyToQMgr
JMS_IBM_MQMD_UserIdentifier .	Identifikátor uživatele	Řetězec	UserIdentifier
JMS_IBM_MQMD_AccountingToken .	Token evidence	Objekt (byte []) [4]	AccountingToken
JMS_IBM_MQMD_ApplIdentityData	Údaje o žádosti vztahující se k totožnosti	Řetězec	ApplIdentityData
JMS_IBM_MQMD_PutApplTyp	Typ aplikace, která vložila zprávu	Celé číslo	PutApplType
Název JMS_IBM_MQMD_PutAppl	Název aplikace, která vložila zprávu	Řetězec	PutApplName
JMS_IBM_MQMD_PutDate .	Datum, kdy byla zpráva vložena	Řetězec	PutDate
JMS_IBM_MQMD_PutTime .	Čas, kdy byla zpráva vložena	Řetězec	PutTime
JMS_IBM_MQMD_ApplOriginData	Údaje o žádosti vztahující se k původu	Řetězec	ApplOriginData
JMS_IBM_MQMD_GroupId	Identifikátor skupiny	Objekt (byte []) [4]	GroupId
JMS_IBM_MQMD_MsgSeqČíslo	Pořadové číslo logické zprávy v rámci skupiny	Celé číslo	MsgSeqNumber
JMS_IBM_MQMD_Offset.	Posunutí dat ve fyzické zprávě od začátku logické zprávy	Celé číslo	Offset
JMS_IBM_MQMD_MsgFlags	Příznaky zprávy	Celé číslo	MsgFlags
JMS_IBM_MQMD_OriginalLength	Délka původní zprávy	Celé číslo	OriginalLength

1.  **Upozornění:** Přiřadíte-li hodnotu JMS_IBM_MQMD_Priority, která není v rozsahu 0-9, porušuje specifikaci JMS .
2.  **Upozornění:** Specifikace JMS uvádí, že ID zprávy musí být nastaveno poskytovatelem JMS a že musí být buď jedinečné, nebo null. Pokud přiřadíte hodnotu k JMS_IBM_MQMD_MsgId, tato hodnota se zkopíruje do JMSMessageID. Není tedy nastaven poskytovatelem JMS a nemusí být jedinečný: jedná se o porušení specifikace JMS .
3.  **Upozornění:** Pokud přiřadíte hodnotu JMS_IBM_MQMD_CorrelId , která začíná řetězcem 'ID:', porušuje specifikaci JMS specifikaci.
4.  **Upozornění:** Použití vlastností bajtového pole na zprávě porušuje specifikaci JMS .

Přístup k datům zprávy produktu IBM MQ z aplikace pomocí produktu IBM MQ classes for JMS
K kompletním datům zprávy produktu IBM MQ můžete přistupovat v rámci aplikace pomocí produktu IBM MQ classes for JMS. Chcete-li přistupovat ke všem datům, musí být tato zpráva JMSBytesMessage. Tělo produktu JMSBytesMessage obsahuje libovolné záhlaví MQRFH2 , všechna ostatní záhlaví IBM MQ a následující data zprávy.

Nastavte vlastnost WMQ_MESSAGE_BODY cíle na WMQ_MESSAGE_BODY_MQ, chcete-li přijmout všechna data těla zprávy v JMSBytesMessage.

Je-li hodnota WMQ_MESSAGE_BODY nastavena na hodnotu WMQ_MESSAGE_BODY_JMS nebo WMQ_MESSAGE_BODY_UNSPECIFIED, tělo zprávy je vráceno bez záhlaví JMS MQRFH2 a vlastnosti produktu JMSBytesMessage odrážejí vlastnosti nastavené v produktu RFH2.

Některé aplikace nemohou používat funkce popsané v tomto tématu. Je-li aplikace připojena ke správci front produktu IBM MQ V6 , nebo pokud byla nastavena hodnota PROVIDERVERSION do produktu 6, funkce nejsou k dispozici.

Odeslání zprávy

Při odesílání zpráv má vlastnost cíle WMQ_MESSAGE_BODYpřednost před hodnotou WMQ_TARGET_CLIENT.

Je-li parametr WMQ_MESSAGE_BODY nastaven na hodnotu WMQ_MESSAGE_BODY_JMS, produkt IBM MQ classes for JMS automaticky vygeneruje záhlaví MQRFH2 na základě nastavení vlastností JMSMessage a polí záhlaví.

Je-li parametr WMQ_MESSAGE_BODY nastaven na hodnotu WMQ_MESSAGE_BODY_MQ, nebude do těla zprávy přidáno žádné další záhlaví.

Je-li parametr WMQ_MESSAGE_BODY nastaven na hodnotu WMQ_MESSAGE_BODY_UNSPECIFIED, produkt IBM MQ classes for JMS odešle záhlaví MQRFH2 , pokud není položka WMQ_TARGET_CLIENT nastavena na hodnotu WMQ_TARGET_DEST_MQ. Při přijetí nastavení WMQ_TARGET_CLIENT na hodnotu WMQ_TARGET_DEST_MQ způsobí, že bude z těla zprávy odebrán některý MQRFH2 .

Poznámka: JMSBytesMessage a JMSTextMessage nevyžadují MQRFH2, zatímco JMSStreamMessage, JMSMapMessage, a JMSObjectMessage .

WMQ_MESSAGE_BODY_UNSPECIFIED je výchozí nastavení pro WMQ_MESSAGE_BODYa WMQ_TARGET_DEST_JMS je výchozí nastavení pro WMQ_TARGET_CLIENT.

Pokud odešlete JMSBytesMessage, můžete potlačit výchozí nastavení pro tělo zprávy produktu JMS , když je vytvořena zpráva IBM MQ . Použijte následující vlastnosti:

- JMS_IBM_Format nebo JMS_IBM_MQMD_Format: Tato vlastnost určuje formát záhlaví IBM MQ nebo informačního obsahu aplikace, který spouští tělo zprávy produktu JMS , pokud neexistuje žádné předchozí záhlaví produktu WebSphere MQ .
- JMS_IBM_Character_Set nebo JMS_IBM_MQMD_CodedCharSetId: Tato vlastnost určuje CCSID záhlaví IBM MQ nebo informačního obsahu aplikace, který spouští tělo zprávy produktu JMS v případě, že neexistuje žádné předchozí záhlaví produktu WebSphere MQ .

- JMS_IBM_Encoding nebo JMS_IBM_MQMD_Encoding: Tato vlastnost určuje kódování záhlaví IBM MQ nebo informačního obsahu aplikace, který spouští tělo zprávy produktu JMS v případě, že neexistuje žádné předchozí záhlaví produktu WebSphere MQ .

Jsou-li zadány oba typy vlastností, potlačí vlastnosti produktu JMS_IBM_MQMD_* odpovídající vlastnosti produktu JMS_IBM_* , pokud je vlastnost cíle WMQ_MQMD_WRITE_ENABLED nastavena na hodnotu true.

Rozdíly v účinku mezi nastavením vlastností zpráv pomocí JMS_IBM_MQMD_* a JMS_IBM_* jsou významné:

1. Vlastnosti JMS_IBM_MQMD_* jsou specifické pro poskytovatele IBM MQ JMS .
2. Vlastnosti JMS_IBM_MQMD_* jsou nastaveny pouze v MQMD. Vlastnosti JMS_IBM_* jsou nastaveny v MQMD pouze v případě, že zpráva nemá záhlaví MQRFH2 JMS . Jinak jsou nastaveny v záhlaví JMS RFH2 .
3. Vlastnosti JMS_IBM_MQMD_* nemají žádný vliv na kódování textu a čísel zapsaných do JMSMessage.

   Přijímající aplikace bude pravděpodobně předpokládat, že hodnoty MQMD.Encoding a MQMD.CodedCharSetId odpovídají kódování a znakové sadě čísel a textu v těle zprávy. Jsou-li použity vlastnosti produktu JMS_IBM_MQMD_* , je odpovědností odesílající aplikace, aby se tak stalo. Kódování a znaková sada čísel a textu v těle zprávy jsou nastaveny vlastnostmi JMS_IBM_* .

   Špatně kódovaný úsek kódu v produktu Obrázek 45 na stránce 216 odesílá zprávu kódovanou ve znakové sadě 1208 s parametrem MQMD.CodedCharSetId nastaveným na hodnotu 37.

   a. Odeslat chybně kódovanou zprávu

   ```
   TextMessage tmo = session.createTextMessage();
   ((MQDestination) destination).setMessageBodyStyle
                   (WMQConstants.WMQ_MESSAGE_BODY_MQ);
   ((MQDestination)destination).setMQMDWriteEnabled(true);
   tmo.setIntProperty(WMQConstants.JMS_IBM_MQMD_CODEDCHARSETID, 37);
   tmo.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET, 1208);
   tmo.setText("String one");
   producer.send(tmo);
   ```

   b. Příjem zprávy závisí na hodnotě parametru JMS_IBM_CHARACTER_SET nastavené hodnotou MQMD.CodedCharSetId:

   ```
   TextMessage tmi = (TextMessage) cons.receive();
   System.out.println("Message is \"" + tmi.getText() + "\"");
   ```

   c. Výsledný výstup:

   ```
   Message is "éÈÊ'>...??>?"
   ```

   Obrázek 45. Nekonzistentně kódováno MQMD a data zprávy

Jedna z úseků kódu v produktu Obrázek 46 na stránce 217 má za následek vložení zprávy do fronty nebo tématu spolu se svým tělem obsahujícím informační obsah aplikace bez automaticky generovaného záhlaví produktu MQRFH2 .

1. Nastavení WMQ_MESSAGE_BODY_MQ:

```
((MQDestination) destination).setMessageBodyStyle
                (WMQConstants.WMQ_MESSAGE_BODY_MQ);
```

2. Nastavení WMQ_TARGET_DEST_MQ:

```
((MQDestination) destination).setMessageBodyStyle
                (WMQConstants.WMQ_MESSAGE_BODY_UNSPECIFIED);
((MQDestination) destination).
                setTargetClient(WMQConstants.WMQ_TARGET_DEST_MQ);
```

Obrázek 46. Odeslat zprávu s tělem zprávy produktu MQ .

Přijímání zprávy

Je-li parametr WMQ_MESSAGE_BODY nastaven na hodnotu WMQ_MESSAGE_BODY_JMS, příchozí a hlavní část zprávy typu JMS se určuje podle obsahu přijaté zprávy produktu WebSphere MQ . Typ zprávy a tělo se určují podle polí v záhlaví MQRFH2 nebo v MQMD, pokud neexistuje MQRFH2.

Je-li parametr WMQ_MESSAGE_BODY nastaven na hodnotu WMQ_MESSAGE_BODY_MQ, typ příchozí zprávy JMS je JMSBytesMessage. Tělo zprávy JMS je data zprávy vrácená základním voláním rozhraní API produktu MQGET . Délka těla zprávy je délka vrácená voláním MQGET . Znaková sada a kódování dat v těle zprávy je určeno poli CodedCharSetId a Encoding v souboru MQMD. Formát dat v těle zprávy je určen polem Formát v poli MQMD .

Je-li parametr WMQ_MESSAGE_BODY nastaven na hodnotu WMQ_MESSAGE_BODY_UNSPECIFIED, nastaví se standardní hodnota IBM MQ classes for JMS na hodnotu WMQ_MESSAGE_BODY_JMS.

Když obdržíte JMSBytesMessage, můžete ji dekódovat pomocí odkazu na následující vlastnosti:

- JMS_IBM_Format nebo JMS_IBM_MQMD_Format: Tato vlastnost určuje formát záhlaví IBM MQ nebo informačního obsahu aplikace, který spouští tělo zprávy produktu JMS , pokud neexistuje žádné předchozí záhlaví produktu WebSphere MQ .
- JMS_IBM_Character_Set nebo JMS_IBM_MQMD_CodedCharSetId: Tato vlastnost určuje CCSID záhlaví IBM MQ nebo informačního obsahu aplikace, který spouští tělo zprávy produktu JMS v případě, že neexistuje žádné předchozí záhlaví produktu WebSphere MQ .
- JMS_IBM_Encoding nebo JMS_IBM_MQMD_Encoding: Tato vlastnost určuje kódování záhlaví IBM MQ nebo informačního obsahu aplikace, který spouští tělo zprávy produktu JMS v případě, že neexistuje žádné předchozí záhlaví produktu WebSphere MQ .

Následující úsek kódu má za následek přijatou zprávu, která je JMSBytesMessage. Bez ohledu na obsah přijaté zprávy a v poli formátu přijaté MQMDse jedná o zprávu JMSBytesMessage.

```
((MQDestination)destination).setMessageBodyStyle
                            (WMQConstants.WMQ_MESSAGE_BODY_MQ);
```

Cílová vlastnost WMQ_MESSAGE_BODY
WMQ_MESSAGE_BODY určuje, zda aplikace JMS zpracovává MQRFH2 zprávy IBM MQ jako součást informačního obsahu zprávy (tedy jako součást těla zprávy produktu JMS).

Tabulka 38. Názvy a popisy vlastností		
Vlastnost	**Krátký formát**	**Popis**
TĚLO WMQ_MESSAGE_BODY	MBODY	Určuje, zda aplikace JMS zpracovává MQRFH2 zprávy IBM MQ jako součást informačního obsahu zprávy (tj. jako součást těla zprávy produktu JMS).

Tabulka 39. Názvy vlastností, hodnoty a metody nastavení			
Vlastnost	**Platné hodnoty v nástroji pro administraci (výchozí nastavení tučně)**	**Platné hodnoty v programech**	**Nastavit metodu**
WMQ_MESSAGE_BODY	• **Neuvedeno** Při odesílání produkt IBM MQ classes for JMS generuje nebo negeneruje a nezahrnuje záhlaví MQRFH2 v závislosti na hodnotě WMQ_TARGET_CLIENT. Je-li příjem, působí jako hodnota JMS. • JMS Při odesílání produkt IBM MQ classes for JMS automaticky vygeneruje záhlaví MQRFH2 a zahrne jej do zprávy produktu IBM MQ . Při příjmu IBM MQ classes for JMS nastavte vlastnosti zprávy JMS podle hodnot v MQRFH2 (pokud existuje). Neprezentuje MQRFH2 jako část těla zprávy JMS . • MQ Při odesílání produkt IBM MQ classes for JMS negeneruje MQRFH2. Při příjmu produkt IBM MQ classes for JMS prezentuje MQRFH2 jako část těla zprávy produktu JMS .	• **WMQ_MESSAGE_ BODY_UNSPECIFIED** • WMQ_MESSAGE_BODY_BODY_J MS • WMQ_MESSAGE_BODY_MQ	setMessageBodySty le

Trvalé zprávy produktu JMS
Aplikace produktu IBM MQ classes for JMS mohou používat atribut fronty produktu **NonPersistentMessageClass** k zajištění lepšího výkonu pro trvalé zprávy produktu JMS , a to na úkor určité spolehlivosti.

Fronta IBM MQ má atribut nazvaný **NonPersistentMessageClass**. Hodnota tohoto atributu určuje, zda jsou přechodné zprávy ve frontě zahozeny, když se správce front restartuje.

Atribut pro lokální frontu můžete nastavit pomocí příkazu IBM MQ Skript (MQSC), DEFINE QLOCAL, s jedním z následujících parametrů:

TŘÍDA NPMCLASS (NORMÁLNÍ)
Netrvalé zprávy ve frontě jsou zahozeny při restartu správce front. Toto je výchozí hodnota.

TŘÍDA NPMCLASS (VYSOKÁ)
Netrvalé zprávy ve frontě se nezařadí, pokud správce front restartuje po uvedení do klidového stavu nebo okamžitého ukončení. Netrvalé zprávy mohou být vyřazeny po preventivním vypnutí nebo selhání.

Toto téma popisuje, jak mohou aplikace produktu IBM MQ classes for JMS používat tento atribut fronty k zajištění lepšího výkonu pro trvalé zprávy produktu JMS .

Vlastnost PERSISTENCE objektu Queue nebo Topic může mít hodnotu HIGH (vysoká). K nastavení této hodnoty můžete použít administrativní nástroj produktu IBM MQ JMS nebo aplikace může volat metodu Destination.setPersistence(), která předává hodnotu parametru WMQConstants.WMQ_PER_NPHIGH jako parametr.

Pokud aplikace odešle trvalou zprávu JMS nebo přechodnou zprávu JMS do místa určení, kde má vlastnost PERSISTENCE hodnotu HIGH, a základní fronta IBM MQ je nastavena na NPMCLASS (HIGH), zpráva bude vložena do fronty jako přechodná zpráva IBM MQ . Pokud vlastnost PERSISTENCE v místě určení nemá hodnotu HIGH nebo pokud je základní fronta nastavena na hodnotu NPMCLASS (NORMAL), je do fronty vložena trvalá zpráva produktu JMS jako trvalá zpráva produktu IBM MQ a do fronty je vložena přechodná zpráva JMS jako přechodná zpráva produktu IBM MQ .

Je-li trvalá zpráva JMS vložena do fronty jako přechodná zpráva IBM MQ a chcete se ujistit, že zpráva nebyla vyřazena po uvedení do klidového stavu nebo okamžitého ukončení práce správce front, musí být všechny fronty, přes které se zpráva směrují, nastaveny na hodnotu NPMCLASS (HIGH). V doméně publikování/odběru tyto fronty zahrnují fronty odběratelů. Jako pomůcce k vynucení této konfigurace vyvolá příkaz IBM MQ classes for JMS výjimku InvalidDestination, pokud se aplikace pokusí vytvořit spotřebitele zpráv pro místo určení, kde vlastnost PERSISTENCE má hodnotu HIGH a základní fronta IBM MQ je nastavena na hodnotu NPMCLASS (NORMAL).

Nastavení vlastnosti PERSISTENCE na místo určení na hodnotu HIGH neovlivní způsob přijetí zprávy z daného místa určení. Zpráva odeslaná jako trvalá zpráva produktu JMS se přijímá jako trvalá zpráva produktu JMS a jako přechodná zpráva produktu JMS je přijata zpráva jako přechodná zpráva JMS .

Když aplikace odešle první zprávu do místa určení, kde má vlastnost PERSISTENCE hodnotu HIGH, nebo když aplikace vytvoří prvního odběratele zpráv pro místo určení, kde má vlastnost PERSISTENCE hodnotu HIGH, IBM MQ classes for JMS vydá volání MQINQ k určení, zda je na základní frontě IBM MQ nastavena hodnota NPMCLASS (HIGH). Žádost musí mít proto oprávnění k zjišťování informací o frontě. Kromě toho produkt IBM MQ classes for JMS zachová výsledek volání MQINQ, dokud nedojde k odstranění místa určení, a nevydá více volání MQINQ. Pokud tedy změníte nastavení NPMCLASS na základní frontě v době, kdy aplikace stále používá místo určení, produkt IBM MQ classes for JMS nové nastavení neoznámí.

Povolíte-li trvalé zprávy produktu JMS do front produktu IBM MQ jako přechodné zprávy produktu IBM MQ , získáte vyšší výkon na úkor určité spolehlivosti. Vyžadujete-li maximální spolehlivost pro trvalé zprávy produktu JMS , neodesílejte zprávy do místa určení, kde má vlastnost PERSISTENCE hodnotu HIGH (vysoká).

Vrstva JMS může používat SYSTEM.JMS.TEMPQ.MODEL, místo SYSTEM.DEFAULT.MODEL.QUEUE. SYSTEM.JMS.TEMPQ.MODEL vytváří trvalé dynamické fronty, které přijímají trvalé zprávy, protože SYSTEM.DEFAULT.MODEL.QUEUE nemůže přijímat trvalé zprávy. Chcete-li použít dočasné fronty k přijímání trvalých zpráv, musíte proto použít SYSTEM.JMS.TEMPQ.MODEL, nebo změňte modelovou frontu na alternativní frontu dle vašeho výběru.

Použití TLS s IBM MQ classes for JMS
Aplikace produktu IBM MQ classes for JMS mohou používat šifrování TLS (Transport Layer Security). K tomu potřebují poskytovatele JSSE.

Připojení produktu IBM MQ classes for JMS používající funkci TRANSPORT (CLIENT) podporují šifrování TLS. TLS poskytuje šifrování komunikace, ověření a integritu zpráv. Obvykle se používá k zabezpečení komunikace mezi dvěma rovnocennými partnery na Internetu nebo v rámci intranetu.

Produkt IBM MQ classes for JMS používá k ošetření šifrování TLS produkt Java Secure Socket Extension (JSSE), a proto vyžaduje poskytovatele JSSE. Prostředí JVM JSE v1.4 má vestavěného poskytovatele JSSE. Podrobnosti o tom, jak spravovat a ukládat certifikáty se mohou lišit od poskytovatele k poskytovateli. Další informace o tomto tématu naleznete v dokumentaci k poskytovateli JSSE.

V tomto oddílu se předpokládá, že poskytovatel JSSE je správně nainstalován a nakonfigurován a že byly nainstalovány a zpřístupněny vhodné certifikáty pro poskytovatele JSSE. Nyní můžete použít JMSAdmin k nastavení počtu administrativních vlastností.

Pokud aplikace IBM MQ classes for JMS používá tabulku definic kanálů klienta (CCDT) k připojení ke správci front, přečtěte si téma "Použití tabulky definic kanálů klienta s IBM MQ classes for JMS" na stránce 252.

Vlastnost objektu SSLCIPHERSUITE
Nastavte hodnotu SSLCIPHERSUITE tak, aby povolujte šifrování TLS v objektu ConnectionFactory .

Chcete-li povolit šifrování TLS v objektu ConnectionFactory , použijte JMSAdmin k nastavení vlastnosti SSLCIPHERSUITE na hodnotu CipherSuite podporovanou vaším poskytovatelem JSSE. To musí odpovídat sadě CipherSpec na cílovém kanálu. Hodnota CipherSuites se však liší od specifikace CipherSpecs , a proto mají různé názvy. Produkt "TLS CipherSpecs a CipherSuites v IBM MQ classes for JMS" na stránce 223 obsahuje tabulku s mapováním CipherSpecs podporovaných produktem IBM MQ na ekvivalentní CipherSuites , jak je známo, na JSSE. Další informace o sadách CipherSpecs a CipherSuites k produktu IBM MQviz Zabezpečení IBM MQ.

Chcete-li například nastavit objekt ConnectionFactory , který lze použít k vytvoření připojení prostřednictvím kanálu MQI s povoleným protokolem TLS se sadou CipherSpec sady TLS_RSA_WITH_AES_128_CBC_SHA, zadejte pro modul JMSAdmin následující příkaz:

```
ALTER CF(my.cf) SSLCIPHERSUITE(SSL_RSA_WITH_AES_128_CBC_SHA)
```

To lze také nastavit z aplikace pomocí metody setSSLCipherSuite () na objektu MQConnectionFactory .

Pokud je pro usnadnění práce zadána vlastnost CipherSpec v rámci vlastnosti SSLCIPHERSUITE, pokusí se služba JMSAdmin mapovat položku CipherSpec na příslušnou sadu CipherSuite a vydá varovnou zprávu. Tento pokus o mapování se nemapuje, je-li vlastnost určena aplikací.

Po prvním použití použijte tabulku CCDT (Client Channel Definition Table). Další informace viz "Použití tabulky definic kanálů klienta s IBM MQ classes for JMS" na stránce 252.

Vlastnost objektu SSLFIPSREQUIRED
Pokud vyžadujete připojení pro použití sady CipherSuite , kterou podporuje poskytovatel prostředí JSSE FIPS produktu IBM Java (IBMJSSEFIPS), nastavte vlastnost SSLFIPSREQUIRED na továrnu připojení na hodnotu YES.

Výchozí hodnota této vlastnosti je NO, což znamená, že připojení může používat jakoukoli sadu CipherSuite , kterou podporuje produkt IBM MQ.

Pokud aplikace používá více než jedno připojení, hodnota SSLFIPSREQUIRED, která se použije, když aplikace vytvoří první připojení, určuje hodnotu, která se použije, když aplikace vytvoří jakékoli následné připojení. To znamená, že hodnota vlastnosti SSLFIPSREQUIRED továrny připojení použitá k vytvoření následného připojení je ignorována. Musíte restartovat aplikaci, chcete-li použít jinou hodnotu SSLFIPSREQUIRED.

Aplikace může tuto vlastnost nastavit voláním metody setSSLFipsRequired () objektu ConnectionFactory . Vlastnost je ignorována, pokud není nastavena žádná sada CipherSuite .

Související informace
Určení, že pro běhové prostředí klienta MQI je použit pouze certifikovaný standard FIPS CipherSpecs
Federální standardy zpracování informací (FIPS) pro UNIX, Linux, and Windows

Vlastnost objektu SSLPEERNAME
Pomocí parametru SSLPEERNAME zadejte vzor rozlišujícího názvu, abyste se ujistili, že se vaše aplikace JMS připojí ke správnému správci front.

Aplikace produktu JMS může zajistit, aby se připojovala ke správnému správci front, a to určením rozlišovacího názvu (DN). Připojení je úspěšné pouze v případě, že správce front představuje DN, které odpovídá vzoru. Další podrobnosti o formátu tohoto vzoru naleznete v souvisejících tématech.

Rozlišující název je nastaven pomocí vlastnosti SSLPEERNAME objektu ConnectionFactory . Například následující příkaz JMSAdmin nastaví objekt ConnectionFactory tak, aby očekával, že se správce front bude identifikovat s názvem Common Name začínajícím znaky `QMGR.` a s alespoň dvěma názvy organizačních jednotek, přičemž první z nich musí být IBM a druhý WEBSPHERE:

```
ALTER CF(my.cf) SSLPEERNAME(CN=QMGR.*, OU=IBM, OU=WEBSPHERE)
```

Při zaškrtnutí se nerozlišují velká a malá písmena a středníky lze použít místo čárek. SSLPEERNAME lze také nastavit z aplikace pomocí metody setSSLPeerName () na objektu MQConnectionFactory . Není-li tato vlastnost nastavena, nebude u rozlišujícího názvu zadaného správcem front provedena žádná kontrola. Tato vlastnost je ignorována, pokud není nastavena žádná sada CipherSuite .

Vlastnost objektu SSLCERTSTORES
Použijte SSLCERTSTORES k uvedení seznamu serverů LDAP, které se mají použít pro kontrolu seznamu odvolaných certifikátů (CRL).

Je běžné použít seznam odvolaných certifikátů (CRL) k identifikaci certifikátů, které již nejsou důvěryhodné. Seznamy CRL jsou obvykle hostovány na serverech LDAP. JMS umožňuje, aby byl server LDAP zadán pro kontrolu CRL pod Java 2 v1.4 nebo novější. Následující příklad JMSAdmin nasměruje produkt JMS na použití seznamu odvolaných certifikátů (CRL) na serveru LDAP s názvem crl1.ibm.com:

```
ALTER CF(my.cf) SSLCRL(ldap://crl1.ibm.com)
```

Poznámka: Chcete-li produkt CertStore úspěšně používat s názvem CRL hostovaným na serveru LDAP, ujistěte se, že vaše sada SDK (Software Development Kit) produktu Java je kompatibilní se seznamem CRL. Některé sady SDK vyžadují, aby seznam CRL odpovídal RFC 2587, které definuje schéma pro protokol LDAP v2. Většina serverů LDAP v3 používá místo toho RFC 2256.

Pokud váš server LDAP není spuštěn na výchozím portu 389, můžete jej zadat připojením dvojtečky (:) a čísla portu k názvu hostitele. Pokud se certifikát prezentovaný správcem front nachází v seznamu odvolaných certifikátů hostovaném na serveru crl1.ibm.com, připojení není dokončeno. Chcete-li se vyhnout jednomu bodu selhání, JMS umožňuje dodat více serverů LDAP tím, že zadáte seznam serverů LDAP oddělených mezerou. Příklad:

```
ALTER CF(my.cf) SSLCRL(ldap://crl1.ibm.com ldap://crl2.ibm.com)
```

Je-li určeno více serverů LDAP, produkt JMS se každé z nich pokouší postupně, dokud nenajde server, na kterém může úspěšně ověřit certifikát správce front. Každý server musí obsahovat stejné informace.

Řetězec v tomto formátu může být dodán aplikací na metodě MQConnectionFactory.setSSLCertStores (). Alternativně může aplikace vytvořit jeden nebo více objektů java.security.cert.CertStore , umístit je do vhodného objektu kolekce a tento objekt kolekce dodat metodě setSSLCertStores (). Tímto způsobem může aplikace upravit kontrolu CRL. Podrobné informace o vytváření a používání objektů CertStore naleznete v dokumentaci k prostředí JSSE.

Certifikát, který předkládá správce front při nastavení připojení, je ověřen následujícím způsobem:

1. První objekt CertStore v kolekci identifikovaný pomocí úložišť sslCertse používá k identifikaci serveru CRL.
2. Došlo k pokusu o kontaktování serveru CRL.
3. Je-li pokus úspěšný, prohledá se server na shodu certifikátu.

a. Pokud má být certifikát odvolán, proces vyhledávání skončil a žádost o připojení selže s kódem příčiny MQRC_SSL_CERTIFICATE_REVOKED.
b. Pokud certifikát nebyl nalezen, je proces vyhledávání znovu spuštěn a připojení je povoleno pokračovat.

4. Je-li pokus o kontaktování serveru neúspěšný, bude použit další objekt CertStore pro identifikaci serveru CRL a proces se opakuje z kroku 2.

   Pokud se jednalo o poslední CertStore v kolekci, nebo pokud kolekce neobsahuje žádné objekty CertStore , došlo k selhání procesu vyhledávání a žádost o připojení se nezdařila s kódem příčiny MQRC_SSL_CERT_STORE_ERROR.

Objekt kolekce určuje pořadí, ve kterém jsou použita položka CertStores .

Pokud vaše aplikace používá úložiště setSSLCertStores () k nastavení kolekce objektů CertStore , nebude již objekt MQConnectionFactory svázán s oborem názvů JNDI. Pokus o to, aby to udělal, způsobuje výjimku. Pokud není nastavena vlastnost sslCertStores, neprovede se kontrola odvolání v certifikátu poskytnutém správcem front. Tato vlastnost je ignorována, pokud není nastavena žádná sada CipherSuite .

Vlastnost objektu SSLRESETCOUNT
Tato vlastnost představuje celkový počet bajtů odeslaných a přijatých připojením před opětovným získáním tajného klíče, který je použit pro šifrování.

Počet odeslaných bajtů je číslo před šifrováním a počet přijatých bajtů je číslo po dešifrování. Počet bajtů obsahuje také řídicí informace odeslané a přijaté produktem IBM MQ classes for JMS.

Chcete-li například konfigurovat objekt ConnectionFactory , který lze použít k vytvoření připojení prostřednictvím kanálu MQI s povoleným zabezpečením TLS s použitím tajného klíče, který je znovu vyjednán po 4 MB dat, zadejte pro správce JMSAdmin následující příkaz:

```
ALTER CF(my.cf) SSLRESETCOUNT(4194304)
```

Aplikace může tuto vlastnost nastavit voláním metody Count () setSSLReset() objektu ConnectionFactory .

Je-li hodnota této vlastnosti nula, což je výchozí hodnota, nebude tajný klíč nikdy znovu vyjednáván. Vlastnost je ignorována, pokud není nastavena žádná sada CipherSuite .

Vlastnost objektu SSLSocketFactory
Chcete-li upravit další aspekty připojení TLS pro aplikaci, vytvořte objekt SSLSocketFactory a nakonfigurujte produkt JMS pro jeho použití.

Možná budete chtít upravit další aspekty připojení TLS pro aplikaci. Například můžete chtít inicializovat kryptografický hardware nebo změnit úložiště klíčů a úložiště údajů o důvěryhodnosti, které používáte. Chcete-li to provést, aplikace musí nejprve vytvořit objekt javax.net.ssl.SSLSocketFactory , který je odpovídajícím způsobem upraven. Informace o tom, jak to provést, jsou uvedeny v dokumentaci k prostředí JSSE, protože přizpůsobitelné funkce se liší od poskytovatele k poskytovateli. Po získání vhodného objektu SSLSocketFactory použijte metodu továrny MQConnectionFactory.setSSLSocketFactory () pro konfiguraci produktu JMS pro použití přizpůsobeného objektu SSLSocketFactory .

Pokud vaše aplikace používá metodu setSSLSocketFactory () k nastavení přizpůsobeného objektu SSLSocketFactory , objekt MQConnectionFactory již nelze svázat s oborem názvů JNDI. Pokus o to, aby to udělal, způsobuje výjimku. Není-li tato vlastnost nastavena, použije se výchozí objekt SSLSocketFactory . Podrobnosti o chování výchozího objektu SSLSocketFactory najdete v dokumentaci k prostředí JSSE. Tato vlastnost je ignorována, pokud není nastavena žádná sada CipherSuite .

Důležité: Nepředpokládejte, že použití vlastností SSL zajišťuje zabezpečení, je-li objekt ConnectionFactory načten z oboru názvů JNDI, který není sám o sobě zabezpečený. Specificky není standardní implementace LDAP rozhraní JNDI zabezpečena. Útočník může imitovat server LDAP, zavádějící aplikaci JMS , aby se připojoval k chybnému serveru, aniž by si toho všimli. Jsou-li zajištěny vhodné mechanismy zabezpečení, jsou zabezpečeny další implementace rozhraní JNDI (jako je implementace fscontext).

Provedení změn v úložišti klíčů nebo úložišti údajů o důvěryhodnosti JSSE
Provedete-li změny v úložišti klíčů nebo úložišti údajů o důvěryhodnosti, musíte provést určité akce, aby změny byly vyzvednuty.

Pokud změníte obsah úložiště klíčů nebo úložiště údajů o důvěryhodnosti prostředí JSSE nebo změníte umístění souboru úložiště klíčů nebo souboru úložiště údajů o důvěryhodnosti, aplikace produktu IBM MQ classes for JMS spuštěné v daném čase automaticky nevezmou tyto změny do paměti. Aby se změny projevily, musí být provedeny následující akce:

- Aplikace musí zavřít všechna svá připojení a zničit veškerá nepoužívaná připojení ve fondech připojení.
- Pokud poskytovatel JSSE ukládá informace z úložiště klíčů a úložiště údajů o důvěryhodnosti do mezipaměti, musí být tyto informace aktualizovány.

Po provedení těchto akcí mohou aplikace znovu vytvořit svá připojení.

V závislosti na návrhu aplikací a na funkci poskytované vaším poskytovatelem JSSE může být možné provést tyto akce bez zastavení a restartování aplikací. Avšak zastavení a restartování aplikací může být nejjednodušším řešením.

TLS CipherSpecs a CipherSuites v IBM MQ classes for JMS
Schopnost aplikací produktu IBM MQ classes for JMS vytvářet připojení ke správci front závisí na specifikaci CipherSpec na konci kanálu MQI a na straně klienta CipherSuite určenou na straně klienta.

V následující tabulce jsou uvedeny seznamy CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites.

Měli byste zkontrolovat téma Zamítnuto CipherSpecs a zjistit, zda některé ze specifikací CipherSpecsuvedené v následující tabulce byly zamítnuty produktem IBM MQ a, pokud ano, které aktualizaci CipherSpec byly zamítnuty.

Důležité: Seznam CipherSuites je seznam podporovaný produktem IBM Java Runtime Environment (JRE) dodávaným s produktem IBM MQ. Seznam CipherSuites , který je uveden, zahrnuje ty, které jsou podporovány prostředím JRE Oracle Java . Další informace o konfiguraci aplikace pro použití prostředí Oracle Java JRE naleznete v tématu Konfigurace aplikace pro použití mapování produktu IBM Java nebo Oracle Java CipherSuite.

Tabulka také uvádí protokol, který se používá pro komunikaci, a zda sada CipherSuite odpovídá standardu FIPS 140-2, či nikoli.

Pokud aplikace nebyla konfigurována k vynucení shody FIPS 140-2, lze použít specifikace Ciphersuites označené jako FIPS 140-2, ale pokud byl pro danou aplikaci nakonfigurován standard FIPS 140-2 (viz následující poznámky v konfiguraci), lze konfigurovat pouze ty CipherSuites , které jsou označeny jako kompatibilní s FIPS 140-2. Při pokusu o použití jiných sad CipherSuites se v chybě zobrazí chybová zpráva.

Poznámka: Každé prostředí JRE může mít více poskytovatelů zabezpečení šifrování, přičemž každý z nich může přispět k implementaci stejné sady CipherSuite. Nicméně ne všichni poskytovatelé zabezpečení jsou certifikováni FIPS 140-2. Pokud není pro aplikaci vynuceno dodržení standardu FIPS 140-2, je možné, že bude použita necertifikovaná implementace sady CipherSuite . Necertifikované implementace nemusí fungovat v souladu se standardem FIPS 140-2, a to ani v případě, že sada CipherSuite teoreticky splňuje minimální úroveň zabezpečení vyžadovanou standardem. Další informace o konfiguraci vynucení FIPS 140-2 v aplikacích produktu IBM MQ JMS naleznete v následujících poznámkách.

Další informace o shodě s FIPS 140-2 a Suite-B pro CipherSpecs a CipherSuitesnajdete v tématu Uvedení CipherSpecs. Možná budete také potřebovat vědět o informacích, které se týkají Federální standardy zpracování informací.

Chcete-li použít úplnou sadu souborů CipherSuites a pracovat s certifikovaným standardem FIPS 140-2 anebo sadou Suite-B, je vyžadováno vhodné prostředí JRE. IBM Produkt Java 7 Service Refresh 4 Fix Pack 2 nebo vyšší úroveň prostředí IBM JRE poskytuje příslušnou podporu.

Poznámka: Chcete-li použít některé CipherSuites, je třeba v prostředí JRE nakonfigurovat 'neomezených' souborů zásad. Další podrobnosti o tom, jak jsou soubory zásad nastaveny v sadě SDK nebo JRE, viz

téma *Soubory zásad sady SDK produktu IBM* v publikaci *Security Reference for IBM SDK, Java Technology Edition* pro verzi, kterou používáte.

Tabulka 40. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
ECDHE_ECDSA_3DES_EDE_CBC_SHA256	SSL_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA	TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA	TLSv1.2	yes

Tabulka 40. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
ECDHE_ECDSA_AES_128_CBC_SHA256	SSL_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	TLSv1.2	yes

Tabulka 40. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
ECDHE_ECDSA_AES_128_GCM_SHA256	SSL_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	TLSv1.2	yes

Tabulka 40. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
ECDHE_ECDSA_AES_256_CBC_SHA384	SSL_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	TLSv1.2	yes

Tabulka 40. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
ECDHE_ECDSA_AES_256_GCM_SHA384	SSL_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	TLSv1.2	yes

Tabulka 40. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
ECDHE_ECDSA_NULL_SHA256	SSL_ECDHE_ECDSA_WITHNULL_SHA	TLS_ECDHE_ECDSA_WITHNULL_SHA	TLSv1.2	ne

Tabulka 40. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
ECDHE_ECDSA_RC4_128_SHA256	SSL_ECDHE_ECDSA_WITH_RC4_128_SHA	TLS_ECDHE_ECDSA_WITH_RC4_128_SHA	TLSv1.2	ne

Tabulka 40. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
ECDHE_RSA_3DES_EDE_CBC_SHA256	SSL_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA	TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA	TLSv1.2	yes

Tabulka 40. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
ECDHE_RSA_AES_128_CBC_SHA256	SSL_ECDHE_RSA_WITH_AES_128_CBC_SHA256	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	TLSv1.2	yes

Tabulka 40. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
ECDHE_RSA_AES_128_GCM_SHA256	SSL_ECDHE_RSA_WITH_AES_128_GCM_SHA256	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	TLSv1.2	yes

Tabulka 40. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
ECDHE_RSA_AES_256_CBC_SHA384	SSL_ECDHE_RSA_WITH_AES_256_CBC_SHA384	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	TLSv1.2	yes

Tabulka 40. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
ECDHE_RSA_AES_256_GCM_SHA384	SSL_ECDHE_RSA_WITH_AES_256_GCM_SHA384	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	TLSv1.2	yes

Tabulka 40. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
ECDHE_RSA_NULL_SHA256	SSL_ECDHE_RSA_WITH_NULL_SHA	TLS_ECDHE_RSA_WITH_NULL_SHA	TLSv1.2	ne

Tabulka 40. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
ECDHE_RSA_RC4_128_SHA256	SSL_ECDHE_RSA_WITH_RC4_128_SHA	TLS_ECDHE_RSA_WITH_RC4_128_SHA	TLSv1.2	ne

Tabulka 40. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
TLS_RSA_WITH_3DES_EDE_CBC_SHA "1" na stránce 247	SSL_RSA_WITH_3DES_EDE_CBC_SHA	TLS_RSA_WITH_3DES_EDE_CBC_SHA	TLSv1	yes

Tabulka 40. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
TLS_RSA_WITH_AES_128_CBC_SHA	SSL_RSA_WITH_AES_128_CBC_SHA	TLS_RSA_WITH_AES_128_CBC_SHA	TLSv1	yes

Tabulka 40. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
TLS_RSA_WITH_AES_128_CBC_SHA256	SSL_RSA_WITH_AES_128_CBC_SHA256	TLS_RSA_WITH_AES_128_CBC_SHA256	TLSv1.2	yes

Tabulka 40. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
TLS_RSA_WITH_AES_128_GCM_SHA256	SSL_RSA_WITH_AES_128_GCM_SHA256	TLS_RSA_WITH_AES_128_GCM_SHA256	TLSv1.2	yes

Tabulka 40. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
TLS_RSA_WITH_AES_256_CBC_SHA	SSL_RSA_WITH_AES_256_CBC_SHA	TLS_RSA_WITH_AES_256_CBC_SHA	TLSv1	yes

Tabulka 40. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
TLS_RSA_WITH_AES_256_CBC_SHA256	SSL_RSA_WITH_AES_256_CBC_SHA256	TLS_RSA_WITH_AES_256_CBC_SHA256	TLSv1.2	yes

Tabulka 40. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
TLS_RSA_WITH_AES_256_GCM_SHA384	SSL_RSA_WITH_AES_256_GCM_SHA384	TLS_RSA_WITH_AES_256_GCM_SHA384	TLSv1.2	yes

Tabulka 40. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
TLS_RSA_WITH_DES_CBC_SHA	SSL_RSA_WITH_DES_CBC_SHA	SSL_RSA_WITH_DES_CBC_SHA	TLSv1	ne

Tabulka 40. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
TLS_RSA_WITH_NULL_SHA256	SSL_RSA_WITH_NULL_SHA256	TLS_RSA_WITH_NULL_SHA256	TLSv1.2	ne

Tabulka 40. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	Ekvivalentní CipherSuite (IBM JRE)	Ekvivalentní CipherSuite (Oracle JRE)	Protokol	Kompatibilní s FIPS 140-2
TLS_RSA_WITH_RC4_128_SHA256	SSL_RSA_WITH_RC4_128_SHA	SSL_RSA_WITH_RC4_128_SHA	TLSv1.2	ne

Notes:

1. Tato CipherSpec TLS_RSA_WITH_3DES_EDE_CBC_SHA byla zamítnuta. Nicméně lze ji přesto použít k přenosu až 32 GB dat před ukončením připojení s chybou AMQ9288. Chcete-li se této chybě vyhnout, je třeba při použití této CipherSpecbuď zabránit použití trojitého DES, nebo povolit resetování tajného klíče.

Konfigurace šifrovacích sad a shody s FIPS v aplikaci IBM MQ classes for JMS

- Aplikace, která používá produkt IBM MQ classes for JMS , může použít jednu ze dvou metod k nastavení CipherSuite pro připojení:
  - Volejte metodu setSSLCipherSuite objektu ConnectionFactory .
  - Použijte administrativní nástroj produktu IBM MQ JMS k nastavení vlastnosti SSLCIPHERSUITE objektu ConnectionFactory .

- Aplikace, která používá produkt IBM MQ classes for JMS , může použít některou ze dvou metod k vynucení shody FIPS 140-2:
  - Volejte metodu setSSLFipsRequired objektu ConnectionFactory .
  - Použijte administrativní nástroj produktu IBM MQ JMS k nastavení vlastnosti SSLFIPSREQUIRED objektu ConnectionFactory .

Konfigurace vaší aplikace pro použití mapování IBM Java nebo Oracle Java CipherSuite

Můžete nakonfigurovat, zda vaše aplikace používá výchozí mapování produktu IBM Java CipherSuite na IBM MQ CipherSpec nebo mapování Oracle CipherSuite na IBM MQ CipherSpec . Z tohoto důvodu můžete použít TLS CipherSuites , zda vaše aplikace používá prostředí IBM JRE nebo Oracle JRE. Vlastnost systému Java `com.ibm.mq.cfg.useIBMCipherMappings` kontroluje, která mapování se používají. Vlastnost může mít jednu z následujících hodnot:

ano
Použijte IBM Java CipherSuite pro mapování IBM MQ CipherSpec .

Tato hodnota je výchozí hodnotou.

ne
Použijte mapování Oracle CipherSuite na IBM MQ CipherSpec .

Další informace o použití IBM MQ Java a šifer TLS naleznete v blogu MQdev MQ Java, TLS Šifry, Non-IBM JRE & APARs IT06775, IV66840, IT09423, IT10837.

Omezení interoperability

Určité CipherSuites mohou být kompatibilní s více než jedním IBM MQ CipherSpec, v závislosti na protokolu, který se používá. však je podporována pouze kombinace CipherSuite/CipherSpec , která používá verzi TLS uvedenou v tabulce 1. Pokus o použití nepodporované kombinace sad CipherSuites a CipherSpecs selže s příslušnou výjimkou. Instalace používající některou z těchto kombinací CipherSuite/CipherSpec by se měly přesunout do podporované kombinace.

Následující tabulka obsahuje sadu CipherSuites , na kterou se vztahuje toto omezení.

Tabulka 41. CipherSuites a jejich podporované a nepodporované CipherSpecs

CipherSuite	**Podporované TLS CipherSpec**	**Nepodporovaná SSL CipherSpec**
SSL_RSA_WITH_3DES_EDE_CBC_SHA	TLS_RSA_WITH_3DES_EDE_CBC_SHA "1" na stránce 248	TRIPLE_DES_SHA_US
SSL_RSA_WITH_DES_CBC_SHA	TLS_RSA_WITH_DES_CBC_SHA	DES_SHA_EXPORT
SSL_RSA_WITH_RC4_128_SHA	TLS_RSA_WITH_RC4_128_SHA256	RC4_SHA_US

Poznámka:

1. Tato CipherSpec TLS_RSA_WITH_3DES_EDE_CBC_SHA byla zamítnuta. Nicméně lze ji přesto použít k přenosu až 32 GB dat před ukončením připojení s chybou AMQ9288. Chcete-li se této chybě vyhnout, je třeba při použití této CipherSpecbuď zabránit použití trojitého DES, nebo povolit resetování tajného klíče.

Zápis kanálu je ukončen v Java pro IBM MQ classes for JMS
Uživatelské procedury kanálu vytvoříte definováním tříd produktu Java , které implementují zadaná rozhraní.

Tři rozhraní jsou definována v balíku com.ibm.mq.exits :

- WMQSendExitpro ukončení odeslání
- WMQReceiveExit, pro uživatelskou proceduru pro příjem

- WMQSecurityExitpro uživatelskou proceduru pro zabezpečení zprávy

Následující ukázka kódu definuje třídu, která implementuje všechna tři rozhraní:

```
public class MyMQExits implements
WMQSendExit, WMQReceiveExit, WMQSecurityExit {
    // Default constructor
  public MyMQExits(){
  }
    // This method implements the send exit interface
  public ByteBuffer channelSendExit(
                                    MQCXP channelExitParms,
                                    MQCD channelDefinition,
                                    ByteBuffer agentBuffer)
  {
    // Complete the body of the send exit here
  }
    // This method implements the receive exit interface
  public ByteBuffer channelReceiveExit(
                                    MQCXP channelExitParms,
                                    MQCD channelDefinition,
                                    ByteBuffer agentBuffer)
  {
    // Complete the body of the receive exit here
  }
    // This method implements the security exit interface
  public ByteBuffer channelSecurityExit(
                                    MQCXP channelExitParms,
                                    MQCD channelDefinition,
                                    ByteBuffer agentBuffer)
  {
    // Complete the body of the security exit here
  }
}
```

Každá uživatelská procedura obdrží jako parametry objekt MQCXP a objekt MQCD. Tyto objekty reprezentují struktury MQCXP a MQCD definované v procedurálním rozhraní.

Je-li zavolána uživatelská procedura pro odeslání zprávy, obsahuje parametr agentBuffer data, která se mají odeslat do správce front serveru. Parametr délky není povinný, protože výraz agentBuffer.limit () poskytuje délku dat. Uživatelská procedura odeslání vrátí jako hodnotu data, která mají být odeslána správci front serveru. Pokud však uživatelská procedura odeslání není poslední uživatelskou procedurou odeslání v posloupnosti uživatelských procedur odeslání, vrátí se vrácená data místo na další uživatelskou proceduru odeslání v posloupnosti. Uživatelská procedura odeslání může vrátit upravenou verzi dat, která přijme, v parametru agentBuffer , nebo může vrátit data nezměněná. Nejjednodušším možným výstupním tělem je proto:

```
{ return agentBuffer; }
```

Je-li volána uživatelská procedura pro přijetí zprávy, obsahuje parametr agentBuffer data, která byla přijata ze správce front serveru. Ukončení příjmu se vrátí jako hodnota, která má být předána do aplikace produktem IBM MQ classes for JMS. Pokud však uživatelská procedura pro přijetí není poslední uživatelskou procedurou příjmu v posloupnosti uživatelských procedur příjmu, vrátí se vrácená data místo na další uživatelskou proceduru pro přijetí v posloupnosti.

Když je zavolána uživatelská procedura zabezpečení, obsahuje parametr agentBuffer data, která byla přijata v toku zabezpečení z uživatelské procedury zabezpečení na konci připojení serveru. Uživatelská procedura zabezpečení se vrátí jako hodnota dat, která mají být odeslána v rámci zabezpečení serveru, do uživatelské procedury zabezpečení serveru.

Uživatelské procedury kanálu jsou volány s vyrovnávací pamětí, která má záložní pole. Pro nejlepší výkon by měla uživatelská procedura vrátit vyrovnávací paměť s záložním polem.

Až bude voláno do uživatelské procedury kanálu, může být předáno až 32 znaků uživatelských dat. Uživatelská procedura přistupuje k datům uživatele voláním metody getExitData () objektu MQCXP. Ačkoli uživatelská procedura může změnit uživatelská data voláním metody setExitData (), uživatelská data se obnoví při každém vyvolání procedury ukončení. Jakékoli změny provedené v uživatelských datech se proto ztratí. Uživatelská procedura však může předávat data z jednoho volání do dalšího pomocí oblasti

uživatelských procedur objektu MQCXP. Uživatelská procedura přistupuje k uživatelské oblasti uživatelské procedury, a to voláním metody getExitUserArea().

Každá výstupní třída musí mít konstruktor. Konstruktorem může být buď výchozí konstruktor, jak je zobrazeno v předchozím příkladu, nebo konstruktor s řetězcovým parametrem. Konstruktor je volán k vytvoření instance třídy ukončení pro každou uživatelskou proceduru definovanou ve třídě. Proto je v předchozím příkladu pro uživatelskou proceduru pro odeslání vytvořena instance třídy MyMQExits , pro uživatelskou proceduru pro příjem je vytvořena jiná instance a pro uživatelskou proceduru zabezpečení je vytvořena třetí instance. Když je volán konstruktor s řetězcovým parametrem, parametr obsahuje stejná uživatelská data, která jsou předána uživatelské proceduře kanálu, pro kterou je instance vytvářena. Pokud má třída ukončení jak výchozí konstruktor, tak konstruktor s jedním parametrem, má přednost konstruktor jednoho parametru.

Nezavírejte připojení z uživatelské procedury kanálu.

Když se data odesílají na konec připojení serveru, šifrování TLS se provádí *po* ukončení kanálu. Podobně platí, že když jsou data přijata ze konce připojení serveru, dešifrování TLS se provádí *před* voláním jakýchkoli uživatelských procedur kanálu.

Ve verzích produktu IBM MQ classes for JMS starších než IBM WebSphere MQ 7.0byly uživatelské procedury kanálu implementovány s použitím rozhraní MQSendExit, MQReceiveExita MQSecurityExit. Tato rozhraní můžete i nadále používat, ale nová rozhraní jsou upřednostňována pro zlepšení funkcí a výkonu.

Konfigurace produktu IBM MQ classes for JMS pro použití uživatelských procedur kanálu
Aplikace produktu IBM MQ classes for JMS může v kanálu MQI, který se spustí při připojení aplikace ke správci front, použít zabezpečení kanálu, odeslání a přijetí. Aplikace může používat uživatelské procedury zapsané v Java, C nebo C + +. Aplikace může také použít posloupnost uživatelských procedur pro odesílání nebo příjem, které jsou spouštěny za dědění.

Následující vlastnosti se používají k určení uživatelské procedury odeslání nebo posloupnosti uživatelských procedur pro odesílání zpráv používaných připojením produktu JMS :

- Vlastnost **SENDEXIT** objektu MQConnectionFactory .
- Vlastnost **sendexit** ve specifikaci aktivace, kterou používá adaptér prostředků IBM MQ pro příchozí komunikaci,
- Vlastnost **sendexit** na objektu ConnectionFactory , kterou používá adaptér prostředků IBM MQ pro výstupní komunikaci.

Hodnota vlastnosti je řetězec, který obsahuje jednu nebo více položek oddělených čárkami. Každá položka identifikuje uživatelskou proceduru pro odeslání zprávy jedním z následujících způsobů:

- Název třídy, která implementuje rozhraní `WMQSendExit` pro uživatelskou proceduru pro odeslání zprávy, která je zapsána v souboru Java.
- Řetězec ve formátu *libraryName* (*entryPointName*) pro uživatelskou proceduru pro odeslání zprávy v jazyce C nebo C + +.

Podobným způsobem určují následující vlastnosti uživatelskou proceduru pro přijetí zprávy nebo posloupnost uživatelských procedur pro příjem, kterou používá připojení:

- Vlastnost **RECEXIT** objektu MQConnectionFactory .
- Vlastnost **receiveexit** ve specifikaci aktivace, kterou používá adaptér prostředků IBM MQ pro příchozí komunikaci,
- Vlastnost **receiveexit** na objektu ConnectionFactory , kterou používá adaptér prostředků IBM MQ pro výstupní komunikaci.

Následující vlastnosti určují proceduru zabezpečení použitou při připojení:

- Vlastnost **SECEXIT** objektu MQConnectionFactory .
- Vlastnost **securityexit** ve specifikaci aktivace, kterou používá adaptér prostředků IBM MQ pro příchozí komunikaci,

- Vlastnost **securityexit** na objektu ConnectionFactory , kterou používá adaptér prostředků IBM MQ pro výstupní komunikaci.

V případě MQConnectionFactoriesmůžete nastavit vlastnosti **SENDEXIT**, **RECEXIT** a **SECEXIT** pomocí nástroje pro administraci produktu IBM MQ JMS nebo IBM MQ Explorer. Alternativně může aplikace nastavit vlastnosti voláním metod `setSendExit()`, `setReceiveExit()`a `setSecurityExit()` .

Uživatelské procedury kanálu jsou načítány vlastním zavaděčem tříd. Chcete-li vyhledat uživatelskou proceduru kanálu, prohledávač tříd prohledá následující umístění v uvedeném pořadí.

1. Cesta ke třídě určená vlastností **com.ibm.mq.cfg.ClientExitPath.JavaExitsClasspath** nebo atributem **JavaExitsClassPath** v sekci Kanály konfiguračního souboru klienta IBM MQ .
2. Cesta ke třídě určená systémovou vlastností Java **com.ibm.mq.exitClasspath**. Mějte na zřeteli, že tato vlastnost je nyní zamítnuta.
3. Adresář IBM MQ opustí adresář, jak ukazuje Tabulka 42 na stránce 251. Zavaděč tříd nejprve prohledá adresář pro soubory tříd, které nejsou zabaleny v souborech archivu produktu Java (JAR). Není-li nalezena uživatelská procedura kanálu, bude zavaděč tříd poté hledat v souborech JAR v adresáři.

Tabulka 42. Adresář uživatelských procedur IBM MQ

Platforma	Adresář
Linux UNIX UNIX and Linux	`/var/mqm/exits` (32bitové uživatelské procedury kanálu) `/var/mqm/exits64` (uživatelské procedury 64bitového kanálu)
Windows Windows	*instalační_dat_adr*\exits kde *instalační_dat_adr* je adresář, který jste vybrali pro datové soubory produktu IBM MQ během instalace. Standardní adresář je C:\ProgramData\IBM\MQ.

Poznámka: Pokud uživatelská procedura kanálu existuje ve více než jednom umístění, načte produkt IBM MQ classes for JMS první nalezenou instanci.

Nadřízeným objektem zavaděče tříd je zaváděč tříd, který se používá k načtení produktu IBM MQ classes for JMS. Je tedy možné, aby nadřízený zavaděč tříd zavedl uživatelskou proceduru kanálu, pokud ji nelze najít v žádném z předchozích umístění. Pokud však používáte produkt IBM MQ classes for JMS v prostředí, jako je například aplikační server JEE , nebudete pravděpodobně moci ovlivnit volbu nadřazeného zavaděče tříd, a proto by měl být zavaděč tříd konfigurován nastavením systémové vlastnosti Java **com.ibm.mq.cfg.ClientExitPath.JavaExitsClasspath** na aplikačním serveru.

Pokud je vaše aplikace spouštěna s povoleným prostorem Java Security Manager , pak konfigurační soubor zásad používaný běhovým prostředím Java , v němž je aplikace spuštěna, musí mít oprávnění k zavedení třídy uživatelské procedury kanálu. Informace o tom, jak to provést, najdete v tématu Spuštění tříd produktu IBM MQ pro aplikace JMS pod produktem Java Security Manager.

Rozhraní `MQSendExit`, `MQReceiveExita` `MQSecurityExit` dodaná s verzemi staršími než IBM WebSphere MQ 7.0 jsou stále podporovány. Pokud používáte uživatelské procedury kanálu, které implementují tato rozhraní, musí být v cestě ke třídě uvedena hodnota `com.ibm.mq.jar` .

Informace o tom, jak zapisovat uživatelské procedury kanálu v jazyce C, najdete v tématu “Kanály-uživatelské programy pro kanály systému zpráv” na stránce 926. Musíte uložit uživatelské programy kanálu napsané v jazycích C nebo C++ v adresáři zobrazeném v produktu Tabulka 42 na stránce 251.

Pokud vaše aplikace používá tabulku definic kanálů klienta (CCDT) k připojení ke správci front, prohlédněte si téma “Použití tabulky definic kanálů klienta s IBM MQ classes for JMS” na stránce 252.

Určení uživatelských dat, která mají být předána uživatelským procedurám kanálu při použití produktu IBM MQ classes for JMS
Až bude voláno do uživatelské procedury kanálu, může být předáno až 32 znaků uživatelských dat.

Vlastnost SENDEXITINIT objektu MQConnectionFactory určuje uživatelská data, která jsou předávána každému ukončovacímu programu odeslání při volání. Hodnota vlastnosti je řetězec, který obsahuje jednu nebo více položek dat uživatele oddělených čárkami. Pozice každé položky uživatelských dat v rámci řetězce určuje, který výstup odeslání bude v posloupnosti uživatelských procedur pro odesílání předán. Například první položka uživatelských dat v řetězci se předá do první uživatelské procedury odeslání v posloupnosti uživatelských procedur odeslání.

Vlastnost SENDEXITINIT můžete nastavit pomocí nástroje pro administraci produktu IBM MQ JMS nebo produktu IBM MQ Explorer. Alternativně může aplikace nastavit tuto vlastnost voláním metody setSendExitInit().

Podobně vlastnost RECEXITINIT objektu ConnectionFactory určuje uživatelská data předávaná pro každou uživatelskou proceduru pro přijetí zprávy a vlastnost SECEXITINIT určuje uživatelská data předávaná uživatelské proceduře zabezpečení. Tyto vlastnosti můžete nastavit pomocí nástroje pro administraci produktu IBM MQ JMS nebo produktu IBM MQ Explorer. Alternativně může aplikace nastavit vlastnosti voláním metod setReceiveExitInit() a setSecurityExitInit().

Všimněte si následujících pravidel, když uvádíte uživatelská data, která jsou předána uživatelským procedurám kanálu:

- Pokud je počet položek uživatelských dat v řetězci více než počet uživatelských procedur v posloupnosti, přebytečné položky uživatelských dat se budou ignorovat.
- Pokud je počet položek uživatelských dat v řetězci menší než počet uživatelských procedur v posloupnosti, každá nespecifikovaná položka uživatelských dat je nastavena na prázdný řetězec. Dva čárky za sebou v řetězci, nebo čárka na začátku řetězce, také označují neuvedenou položku uživatelských dat.

Pokud aplikace používá tabulku CCDT (Client Channel Definition CCDT) k připojení ke správci front, budou veškerá uživatelská data zadaná v definici kanálu připojení klienta předána uživatelským procedurám kanálu při jejich volání. Další informace o použití tabulky definic kanálů klienta viz “Použití tabulky definic kanálů klienta s IBM MQ classes for JMS” na stránce 252.

Použití tabulky definic kanálů klienta s IBM MQ classes for JMS
Aplikace produktu IBM MQ classes for JMS může používat definice kanálů připojení klienta, které jsou uloženy v tabulce CCDT (Client Channel Definition table). Objekt ConnectionFactory nakonfigurujete pro použití tabulky CCDT. Existují některá omezení jejího používání.

Jako alternativu k vytvoření definice kanálu připojení klienta pomocí nastavení určitých vlastností objektu ConnectionFactory může aplikace IBM MQ classes for JMS použít definice kanálů připojení klienta, které jsou uloženy v tabulce definic kanálů klienta. Tyto definice jsou vytvořeny příkazy skriptu IBM MQ Script (MQSC) nebo příkazy PCF (IBM MQ Programmable Command Format). Když aplikace vytvoří objekt připojení, produkt IBM MQ classes for JMS prohledá tabulku definic kanálů klienta pro vhodnou definici kanálu připojení klienta a použije definici kanálu ke spuštění kanálu MQI. Další informace o tabulkách definic kanálů klienta a o tom, jak je vytvořit, najdete v tématu Tabulka definic kanálů klienta.

Chcete-li použít tabulku definic kanálů klienta, vlastnost CCDTURL objektu ConnectionFactory musí být nastavena na objekt adresy URL. IBM MQ classes for JMS nečte informace o tabulce CCDT z konfiguračního souboru produktu IBM MQ MQI client , ačkoli jsou zde použity některé jiné hodnoty (viz “Konfigurační soubor IBM MQ classes for JMS” na stránce 81 , pro které je použita hodnota). Objekt URL zapouzdřuje adresu URL, která identifikuje název a umístění souboru obsahujícího tabulku definic kanálů klienta a určuje, jak lze k souboru přistupovat. Vlastnost CCDTURL můžete nastavit pomocí nástroje pro administraci produktu IBM MQ JMS nebo aplikaci může nastavit vlastnost tak, že vytvoří objekt URL a zavolá metodu setCCDTURL() objektu ConnectionFactory .

Pokud například soubor ccdt1.tab obsahuje tabulku definic kanálů klienta a je uložen ve stejném systému, v němž je aplikace spuštěna, může aplikace nastavit vlastnost CCDTURL následujícím způsobem:

```
java.net.URL chanTab1 = new URL("file:///home/admdata/ccdt1.tab");
factory.setCCDTURL(chanTab1);
```

Jako další příklad předpokládejme, že soubor ccdt2.tab obsahuje tabulku definic kanálů klienta a je uložen v systému, který se liší od tabulky, na které je aplikace spuštěna. Je-li k souboru přístup pomocí protokolu FTP, může aplikace nastavit vlastnost CCDTURL následujícím způsobem:

```
java.net.URL chanTab2 = new URL("ftp://ftp.server/admdata/ccdt2.tab");
factory.setCCDTURL(chanTab2);
```

Kromě nastavení vlastnosti CCDTURL objektu ConnectionFactory , musí být vlastnost QMANAGER stejného objektu nastavena na jednu z následujících hodnot:

- Název správce front
- Hvězdička (*) následovaná názvem skupiny správců front

Jedná se o tytéž hodnoty, které lze použít pro parametr **QMgrName** v rámci volání MQCONN vydaného aplikací klienta, která používá rozhraní MQI (Message Queue Interface). Další informace o významu těchto hodnot najdete v tématu MQCONN. Vlastnost QMANAGER můžete nastavit pomocí nástroje pro administraci produktu IBM MQ JMS nebo Průzkumníka IBM MQ . Alternativně může aplikace nastavit vlastnost voláním metody setQueueManager () objektu ConnectionFactory .

Pokud aplikace poté vytvoří objekt připojení z objektu ConnectionFactory , produkt IBM MQ classes for JMS přistoupí k tabulce definic kanálů klienta identifikované vlastností CCDTURL, použije vlastnost QMANAGER k prohledání tabulky pro vhodnou definici kanálu připojení klienta a poté použije definici kanálu ke spuštění kanálu MQI pro správce front.

Všimněte si, že vlastnosti CCDTURL a CHANNEL objektu ConnectionFactory nemohou být nastaveny, když aplikace volá metodu createConnection(). Jsou-li nastaveny obě vlastnosti, metoda vygeneruje výjimku. Vlastnost CCDTURL nebo CHANNEL se považuje za sadu, je-li její hodnota jakákoli jiná než null, prázdný řetězec nebo řetězec obsahující všechny prázdné znaky.

Když produkt IBM MQ classes for JMS najde vhodnou definici kanálu pro připojení klienta v tabulce definic kanálů klienta, použije pouze informace extrahované z tabulky ke spuštění kanálu MQI. Všechny vlastnosti související s kanálem objektu ConnectionFactory se budou ignorovat.

Zejména si povšimněte následujících bodů, pokud používáte TLS:

- Kanál MQI používá zabezpečení TLS pouze v případě, že definice kanálu extrahovaná z definiční tabulky kanálu klienta určuje název CipherSpec podporovaný produktem IBM MQ classes for JMS.
- Tabulka definic kanálů klienta také obsahuje informace o umístění serverů LDAP (Lightweight Directory Access Protocol), které uchovávají seznamy zrušených certifikátů (CRL). Produkt IBM MQ classes for JMS používá pouze tyto informace pro přístup k serverům LDAP, které obsahují seznamy odvolaných certifikátů (CRL).
- Definiční tabulka kanálu klienta může také obsahovat umístění odpovídacího modulu OCSP. Produkt IBM MQ classes for JMS nemůže používat informace OCSP v souboru s tabulkou definic kanálů klienta. Nicméně můžete OCSP nakonfigurovat tak, jak je popsáno v sekci Protokol OCSP (Online Certificate Status Protocol) v aplikacích klienta Java a JMS.

Další informace o použití protokolu TLS s tabulkou definic kanálů klienta naleznete v tématu Použití rozšířeného transakčního klienta s kanály TLS.

Všimněte si také následujících bodů, pokud používáte uživatelské procedury kanálu:

- Kanál MQI používá pouze uživatelské procedury kanálu a přidružená uživatelská data určená definicí kanálu extrahovanou z tabulky definic kanálů klienta.
- Definice kanálu extrahovaná z tabulky definic kanálů klienta může určovat uživatelské procedury kanálu, které jsou zapsány v produktu Java. To znamená například, že parametr SCYEXIT v příkazu DEFINE CHANNEL k vytvoření definice kanálu připojení klienta může určovat název třídy, která implementuje rozhraní WMQSecurityExit . Podobně může parametr SENDEXIT zadat název třídy, která implementuje rozhraní WMQSendExit , a parametr RCVEXIT může uvádět název třídy, která implementuje rozhraní WMQReceiveExit . Další informace o tom, jak zapisovat uživatelskou proceduru kanálu v produktu Java, viz“Zápis kanálu je ukončen v Java pro IBM MQ classes for JMS” na stránce 248.

Použití uživatelských procedur kanálu napsaných v jiném jazyce, než je Java , je podporováno také. Informace o tom, jak určit parametry SCYEXIT, SENDEXIT a RCVEXIT v příkazu DEFINE CHANNEL pro uživatelské procedury kanálu zapsané v jiném jazyce, najdete v tématu DEFINE CHANNEL.

Automatické opětovné připojení klienta JMS
Konfigurujte klienta produktu JMS , aby se znovu připojil automaticky po síti, správci front nebo selhání serveru.

V případě, že je samostatná aplikace IBM MQ classes for JMS připojena ke správci front pomocí přenosu klienta a správce front se z nějakého důvodu stane nedostupným (kvůli výpadku sítě, selhání správce front nebo zastavenému správci front), modul IBM MQ classes for JMS vygeneruje výjimku JMSException při příštím pokusu aplikace o komunikaci se správcem front. Aplikace musí zachytit výjimku JMSException a pokusit se znovu připojit ke správci front. Návrh aplikace můžete zjednodušit tak, že povolíte automatické opětovné připojení klienta. Když se správce front stane nedostupným, pokusí se produkt IBM MQ classes for JMS automaticky znovu připojit ke správci front jménem aplikace. To znamená, že aplikace nemusí obsahovat logiku pro nové připojení.

Automatické opětovné připojení klienta je dostupné pouze pro samostatné aplikace IBM MQ classes for JMS . Použití automatického opětovného připojení klienta v rámci platformy Java Platform, Enterprise Edition není podporováno.

Automatické opětovné připojení klienta JMS pomocí CONNECTIONNAMELIST
Pokud samostatná aplikace IBM MQ classes for JMS používá továrnu připojení, která má sadu vlastností CONNECTIONNAMELIST, je aplikace způsobilá k použití automatického připojení klienta.

Chování funkcí automatického opětovného připojení klienta, které poskytuje produkt IBM MQ classes for JMS , závisí na vlastnostech, které následují:

Vlastnost TRANSPORT továrny připojení produktu JMS (Krátký název TRAN)

Metoda TRANSPORT určuje způsob připojení aplikací, které používají továrnu připojení, ke správci front. Tato vlastnost musí být nastavena na hodnotu CLIENT pro automatické opětovné připojení klienta, které má být použito. Automatické opětovné připojení klienta není dostupné pro aplikace, které se připojují ke správci front, který používá továrnu připojení, která má vlastnost TRANSPORT nastavenou na BIND, DIRECT nebo DIRECTHTTP.

Vlastnost továrny připojení QMANAGER produktu JMS (Krátký název QMGR)

Vlastnost QMANAGER uvádí název správce front, ke kterému se továrna připojení připojuje.

Vlastnost továrny připojení CONNECTIONNAMELIST produktu JMS (Krátký název CRHOSTS)

Vlastnost CONNECTIONNAMELIST je seznam oddělený čárkami, kde každý záznam obsahuje informace o názvu hostitele a portu, které mají být použity pro připojení ke správci front uvedenému vlastností QMANAGER, když používáte přenos CLIENT. Seznam má následující formát: název hostitele (port), název hostitele (port).

Vlastnost továrny připojení CLIENTRECONNECTOPTIONS produktu JMS Connection Factory (Krátký název CROPT)

CLIENTRECONNECTOPTIONS řídí, zda se produkt IBM MQ classes for JMS pokusí automaticky připojit ke správci front jménem aplikace, pokud je správce front k dispozici.

Atribut DefRecon v sekci Kanály konfiguračního souboru klienta

Atribut DefRecon poskytuje administrativní volbu, která umožňuje všem aplikacím automaticky znovu navázat připojení nebo zakázat automatické opětovné připojení pro aplikace, které jsou automaticky znovu připojované k opětovnému připojení.

Automatické opětovné připojení klienta je dostupné pouze tehdy, když se aplikace úspěšně připojí ke správci front.

Když se aplikace připojí ke správci front, který používá přenos CLIENT, použije produkt IBM MQ classes for JMS hodnotu vlastnosti továrny připojení CLIENTRECONNCONNECTOPTIONS k určení, zda má být použito automatické opětovné připojení klienta, pokud je správce front, k němuž je aplikace připojena,

nedostupný. Tabulka 1 uvádí možné hodnoty vlastnosti CLIENTRECONECOPTIONS a chování IBM MQ classes for JMS pro každou z těchto hodnot:

Tabulka 43. Možné hodnoty vlastnosti CLIENTRECCECTOPTIONS.

CLIENTRECONNECTOPTIONS	**Chování objektu IBM MQ classes for JMS**
ANY	Použijte hodnotu vlastnosti CONNECTIONNAMELIST k otevření připojení k kombinaci názvu hostitele a portu a připojte se k libovolnému správci front. Chcete-li použít tuto volbu automatického opětovného připojení klienta, vlastnost QMANAGER musí být nastavena na výchozí hodnotu nebo "*".
VZEZ.	Použijte hodnotu parametru DefRecon , abyste určili, zda je automatické opětovné připojení klienta k dispozici.
VYPNUTO	Neprovádějte žádné automatické opětovné připojení klienta a nevracejte výjimku JMSException do aplikace.
QMGR	Použijte hodnotu vlastnosti CONNECTIONNAMELIST k otevření připojení k kombinaci názvu hostitele a portu a připojte se ke správci front uvedenému ve vlastnosti QMANAGER.

Když provádíte automatické opětovné připojení klienta, IBM MQ classes for JMS použije informace v vlastnosti ConnectionFactory CONNNECTIONNAMELIST k určení, ke kterému systému se má znovu připojit.

IBM MQ classes for JMS se nejprve pokusí znovu připojit pomocí názvu hostitele a portu uvedeného v první položce v CONNECTIONNAMELIST. Je-li vytvořeno připojení, produkt IBM MQ classes for JMS se pak pokusí připojit ke správci front, který má název uvedený ve vlastnosti QMANAGER. Pokud lze navázat připojení ke správci front, program IBM MQ classes for JMS znovu otevře všechny objekty produktu IBM MQ , které měla aplikace otevřená před automatickým opětovným připojením klienta, a pokračovat v práci jako předtím.

Pokud nelze vytvořit připojení k požadovanému správci front pomocí první položky v CONNECTIONNAMELIST, pokusí se produkt IBM MQ classes for JMS druhou položku v CONNECTIONNAMELIST a tak dále.

Když příkaz IBM MQ classes for JMS vyzkoušel všechny položky v CONNECTIONNAMELIST, čekají po určitou dobu, než se znovu pokusí znovu navázat spojení. Chcete-li provést nový pokus o nový pokus o připojení, spustí se IBM MQ classes for JMS prvním záznamem v CONNECTIONNAMELIST. Poté vyzkoušejte každou položku v CONNECTIONNAMELIST postupně, dokud nedojde k opětovnému připojení nebo dokud není dosaženo konce CONNECTIONNAMELIST, v tom případě IBM MQ classes for JMS čeká po určitou dobu, než se pokusí znovu.

Tento proces automatického opětovného připojení klienta pokračuje, dokud se produkt IBM MQ classes for JMS úspěšně znovu nepřipojí ke správci front určenému vlastností QMANAGER.

Při výchozím nastavení dojde k pokusům o opětovné připojení v následujících intervalech:

- První pokus se provede po počátečním zpoždění o 1 sekundu plus náhodný prvek až do 250 milisekund.
- Druhý pokus se provede 2 sekundy s náhodným intervalem až 500 milisekund, po selhání prvního pokusu.
- Třetí pokus se provede 4 sekundy s náhodným intervalem do 1 sekundy po selhání druhého pokusu.
- Čtvrtý pokus se provede o 8 sekund s náhodným intervalem do 2 sekund, po třetím pokusu selže.

- Pátý pokus je proveden 16 sekund a náhodný interval až 4 sekundy po selhání čtvrtého pokusu o selhání.
- Šestý pokus a všechny následné pokusy jsou provedeny 25 sekund s náhodným intervalem po dobu 6 sekund a 250 milisekund po selhání předchozího pokusu.

Pokusy o opětovné připojení jsou zpožděny o intervaly, které jsou částečně fixní a částečně náhodné. Tím zabráníte všem aplikacím produktu IBM MQ classes for JMS , které byly připojeny ke správci front, který již není k dispozici pro nové připojení současně.

Potřebujete-li zvýšit výchozí hodnoty, aby přesněji odrážely dobu potřebnou pro zotavení správce front nebo pohotovostní správce front, který má být aktivní, upravte atribut ReconDelay v sekci Channel konfiguračního souboru klienta, abyste získali další informace, viz stanza CHANNELS v konfiguračním souboru klienta.

Zda aplikace IBM MQ classes for JMS pracuje správně po opětovném připojení automaticky, závisí na jejím návrhu. Přečtěte si související témata, abyste porozuměli tomu, jak navrhnout aplikace mohou využívat funkce automatického opětovného připojení.

Připojení ke správcům front s více instancemi pomocí CONNECTIONNAMELIST
Automatické opětovné připojení klienta může být používáno aplikacemi produktu IBM MQ classes for JMS , které se připojují ke správci front s více instancemi.

Pokud se instance správce front, kterou aplikace používá, stane nedostupnou, může se produkt IBM MQ classes for JMS automaticky pokusit o připojení k instanci v pohotovostním režimu v zastoupení aplikace. Bloky aplikací, když probíhá automatické opětovné připojení klienta, a obnoví se, když produkt IBM MQ classes for JMS naváže spojení se správcem front v pohotovostním režimu.

Chcete-li povolit automatické opětovné připojení klienta pro správce front s více instancemi, nastavte následující vlastnosti v Továrně připojení, které používá aplikace IBM MQ classes for JMS :

CHANNEL
Název kanálu připojení serveru definovaného ve správci front.

QMANAGER

Název správce front s více instancemi.

CONNECTIONNAMELIST=host1(port1), host2(port2).

První položka v seznamu musí obsahovat název hostitele a port, který se používá ke kontaktování primární instance správce front. Druhá položka by měla obsahovat název hostitele a port systému, na kterém je umístěna instance správce front v pohotovostním režimu.

KLIENTCONNECTIONS=QMGR.

Tím je zajištěno, že se produkt IBM MQ classes for JMS pokusí znovu připojit ke správci front se stejným názvem jako správce front, ke kterému byla aplikace dříve připojena.

Automatické opětovné připojení klienta JMS k CCDTs
Pokud samostatná aplikace IBM MQ classes for JMS používá továrnu připojení, která má sadu vlastností CCDTURL, je aplikace vhodná k použití automatického opětovného připojení klienta.

Chování funkcí automatického opětovného připojení klienta, které poskytuje produkt IBM MQ classes for JMS , závisí na následujících vlastnostech:

Vlastnost TRANSPORT továrny připojení produktu JMS (Krátký název TRAN)

Metoda TRANSPORT určuje způsob připojení aplikací, které používají továrnu připojení, ke správci front. Tato vlastnost musí být nastavena na hodnotu CLIENT pro automatické opětovné připojení klienta, které má být použito. Automatické opětovné připojení klienta není k dispozici pro aplikace, které se připojují ke správci front pomocí továrny připojení, která má vlastnost TRANSPORT nastavenou na BIND, DIRECT nebo DIRECTHTTP.

Vlastnost továrny připojení QMANAGER produktu JMS (Krátký název QMGR)

Vlastnost QMANAGER uvádí název správce front, ke kterému se továrna připojení připojuje.

Vlastnost CCDTURL továrny připojení produktu JMS CCDTURL (Short name CCDT)

Vlastnost CCDTURL ukazuje na tabulku definic kanálů klienta, kterou produkt IBM MQ classes for JMS používá při připojování ke správci front.

Vlastnost továrny připojení CLIENTRECONNECTOPTIONS produktu JMS Connection Factory (Krátký název CROPT)

CLIENTRECONNECTOPTIONS řídí, zda se produkt IBM MQ classes for JMS pokusí automaticky připojit ke správci front jménem aplikace, pokud je správce front k dispozici.

Atribut DefRecon v sekci Kanály konfiguračního souboru klienta

Atribut DefRecon poskytuje administrativní volbu, která umožňuje všem aplikacím automaticky znovu navázat připojení nebo zakázat automatické opětovné připojení pro aplikace, které jsou automaticky znovu připojované k opětovnému připojení.

Automatické opětovné připojení klienta je dostupné pouze tehdy, když se aplikace úspěšně připojí ke správci front.

Když se aplikace připojí ke správci front pomocí přenosu CLIENT, použije produkt IBM MQ classes for JMS hodnotu vlastnosti továrny připojení CLIENTRECONECOPTIONS k určení, zda má být použito automatické opětovné připojení klienta, pokud je správce front, k němuž je aplikace připojena, k dispozici. Tabulka 1 uvádí možné hodnoty vlastnosti CLIENTRECONECOPTIONS a chování IBM MQ classes for JMS pro každou z těchto hodnot:

Tabulka 44. Možné vlastnosti CLIENTRECCECTOPTIONS

CLIENTRECONNECTOPTIONS	**Chování objektu IBM MQ classes for JMS**
ANY	Otevřete tabulku definic kanálů klienta, která je určena vlastností CCDTURL, vyberte položku v tabulce a poté ji použijte ke spuštění kanálu připojení klienta ke správci front. Chcete-li použít tuto volbu automatického nového připojení klienta, vlastnost QMANAGER musí být nastavena na: • Hvězdička (*) • Hvězdička (*) následovaná názvem skupiny správců front • Prázdný řetězec nebo řetězec, který obsahuje všechny prázdné znaky
VZEZ.	Použijte hodnotu parametru DefRecon , abyste určili, zda je automatické opětovné připojení klienta k dispozici.
VYPNUTO	Neprovádějte žádné automatické opětovné připojení klienta a nevracejte výjimku JMSException do aplikace.
QMGR	Otevřete tabulku definic kanálů klienta, která je určena vlastností CCDTURL, najít položky v tabulce, které odpovídají názvu správce front určenému vlastností QMANAGER, a poté tyto položky použít ke spuštění kanálu připojení klienta k tomuto správci front.

Při provádění automatického opětovného připojení klienta používá produkt IBM MQ classes for JMS tabulku definic kanálů klienta, která je uvedena ve vlastnosti CCDTURL, k určení systému, ke kterému se má znovu připojit.

Produkt IBM MQ classes for JMS nejprve analyzuje tabulku definic kanálů klienta a nalezne vhodnou položku, která odpovídá hodnotě vlastnosti QMANAGER. Když je nalezena položka, IBM MQ classes for JMS se pokusí znovu připojit k požadovanému správci front pomocí této položky. Pokud lze navázat připojení ke správci front, program IBM MQ classes for JMS znovu otevře všechny objekty produktu IBM MQ , které měla aplikace otevřená před automatickým opětovným připojením klienta, a pokračovat v práci jako předtím.

Pokud nelze navázat spojení s požadovaným správcem front, IBM MQ classes for JMS hledá další vhodnou položku v tabulce definic kanálů klienta a pokusí se ji použít atd.

Když se IBM MQ classes for JMS pokusilo o všechny vhodné položky v tabulce definic kanálů klienta, čekají po určitou dobu, než se znovu pokusí znovu navázat spojení. Chcete-li provést nový pokus o opětovné připojení, program IBM MQ classes for JMS znovu analyzuje tabulku definic kanálů klienta a pokusí se o první vhodnou položku. Poté zkusí všechny vhodné položky v tabulce definic kanálů klienta znovu, dokud nedojde k opětovnému připojení nebo se pokusíte o poslední vhodnou položku v tabulce definic kanálů klienta, v tom případě IBM MQ classes for JMS čeká po určitou dobu a poté se znovu pokusí.

Tento proces automatického opětovného připojení klienta pokračuje, dokud se produkt IBM MQ classes for JMS úspěšně znovu nepřipojí ke správci front určenému vlastností QMANAGER.

Při výchozím nastavení dojde k pokusům o opětovné připojení v následujících intervalech:

- První pokus se provede po počátečním zpoždění o 1 sekundu plus náhodný prvek až do 250 milisekund.
- Druhý pokus se provede 2 sekundy s náhodným intervalem až 500 milisekund, po selhání prvního pokusu.
- Třetí pokus se provede 4 sekundy s náhodným intervalem do 1 sekundy po selhání druhého pokusu.
- Čtvrtý pokus se provede o 8 sekund s náhodným intervalem do 2 sekund, po třetím pokusu selže.
- Pátý pokus je proveden 16 sekund a náhodný interval až 4 sekundy po selhání čtvrtého pokusu o selhání.
- Šestý pokus a všechny následné pokusy jsou provedeny 25 sekund s náhodným intervalem po dobu 6 sekund a 250 milisekund po selhání předchozího pokusu.

Pokusy o opětovné připojení jsou zpožděny o intervaly, které jsou částečně fixní a částečně náhodné. Tím se zabrání všem aplikacím produktu IBM MQ classes for JMS , které byly připojeny ke správci front, který již není k dispozici, aby se znovu připojovaly současně.

Potřebujete-li zvýšit výchozí hodnoty, aby přesněji odrážely dobu potřebnou pro zotavení správce front nebo pohotovostní správce front, který má být aktivní, upravte atribut ReconDelay v sekci Channel konfiguračního souboru klienta, abyste získali další informace, viz stanza CHANNELS v konfiguračním souboru klienta.

To, zda aplikace IBM MQ classes for JMS pracuje správně po opětovném připojení automaticky, závisí na jeho návrhu. Přečtěte si související témata, abyste porozuměli tomu, jak navrhnout aplikace, které mohou využívat funkce automatického opětovného připojení.

Připojení ke správcům front s více instancemi pomocí CCDT
Automatické opětovné připojení klienta může být používáno aplikacemi produktu IBM MQ classes for JMS , které se připojují ke správci front s více instancemi.

Pokud se instance správce front, kterou aplikace používá, stane nedostupnou, může se produkt IBM MQ classes for JMS automaticky pokusit o připojení k instanci v pohotovostním režimu v zastoupení aplikace. Bloky aplikací, když probíhá automatické opětovné připojení klienta, a obnoví se, jakmile produkt IBM MQ classes for JMS vytvoří připojení ke správci front v pohotovostním režimu.

Chcete-li povolit automatické opětovné připojení klienta pro správce front s více instancemi, nastavte následující vlastnosti v Továrně připojení, které používá aplikace IBM MQ classes for JMS :

QMANAGER = Název správce front s více instancemi.

CCDTURL=URI

Identifikátor URI pro tabulku definic kanálů klienta, která obsahuje dva záznamy pro správce front s více instancemi; jeden pro primární instanci a jeden pro instanci podle instance.

Použití automatického opětovného připojení klienta v prostředí Java SE a Java EE
Informace o tom, jak používat automatické opětovné připojení klienta IBM MQ a správce front s více instancemi, v prostředí Java SE a Java EE .

Správci front s více instancemi jsou instance stejného správce front konfigurovaného na různých serverech. Jedna instance správce front je definována jako aktivní instance a jiná instance je definována jako instance v pohotovostním režimu. Dojde-li k selhání aktivní instance, správce front s více instancemi se automaticky restartuje na záložním serveru.

Aktivní i záložní správci front mají stejný identifikátor správce front (QMID). Klientské aplikace produktu IBM MQ , které se připojují ke správci front s více instancemi, lze nakonfigurovat tak, aby se automaticky znovu připojovaly k instanci v pohotovostním režimu správce front s použitím automatického opětovného připojení klienta.

Související informace
Správci front s více instancemi
Automatické opětovné připojení klienta

Použití automatického opětovného připojení klienta v prostředí Java SE
Aplikace používající produkt IBM MQ classes for JMS běžící v prostředí Java SE mohou využívat funkce automatického opětného připojení klienta prostřednictvím vlastnosti továrny připojení **CLIENTRECONNECTOPTIONS**.

Vlastnost továrny připojení **CLIENTRECONNECTOPTIONS**, která je k dispozici v produktu IBM WebSphere MQ 7.0.1 Fix Pack 3 a vyšší, používá dvě další vlastnosti továrny připojení, **CONNECTIONNAMELIST** a **CCDTURL**k určení způsobu připojení k serveru, na kterém je správce front spuštěn.

CONNECTIONNAMELIST vlastnost

Vlastnost **CONNECTIONNAMELIST** je seznam oddělený čárkami, který obsahuje informace o názvu hostitele a portu, které mají být použity pro připojení ke správci front v režimu klienta. Tato vlastnost se používá spolu s hodnotami **QMANAGER** a **CHANNEL** . Když aplikace používá vlastnost **CONNECTIONNAMELIST** k vytvoření připojení klienta, pokusí se IBM MQ classes for JMS o připojení k jednotlivým hostitelům v pořadí seznamu. Je-li první hostitel správce front nedostupný, pokusí se příkaz IBM MQ classes for JMS o připojení k dalšímu hostiteli v seznamu. Je-li dosažen konec seznamu názvů připojení bez vytvoření připojení, IBM MQ classes for JMS vygeneruje kód příčiny `MQRC_QMGR_NOT_AVAILABLE` IBM MQ .

If the queue manager that the application is connected to fails, any applications that used a **CONNECTIONNAMELIST** to connect to that queue manager receive an exception indicating the queue manager is not available. Aplikace musí zachytit výjimku a vymazat všechny prostředky, které používala. Chcete-li vytvořit připojení, aplikace musí použít továrnu připojení. Továrna připojení se pokouší znovu připojit ke každému hostiteli v pořadí seznamu, správce front, který selhal, není nyní k dispozici. Továrna připojení se pokusí připojit k jinému hostiteli v seznamu.

CCDTURL vlastnost

Vlastnost **CCDTURL** obsahuje adresu URL (Uniform Resource Locator), která ukazuje na tabulku CCDT (Client Channel Definition Table), tato vlastnost se používá spolu s vlastností **QMANAGER** . Tabulka CCDT obsahuje seznam kanálů klienta, které se používají pro připojení ke správci front definovanému v systému IBM MQ . Informace o tom, jak CCDT používá IBM MQ classes for JMS, viz Použití tabulky definic kanálů klienta s třídami produktu IBM MQ pro platformu JMS.

Použití vlastnosti CLIENTRECONNECTOPTIONS k povolení automatického opětovného připojení klienta v rámci IBM MQ classes for JMS

Vlastnost **CLIENTRECONNECTOPTIONS** se používá k povolení automatického opětovného připojení klienta v rámci IBM MQ classes for JMS. Možné hodnoty pro tuto vlastnost jsou následující:

ASDEF
Chování automatického opětovného připojení klienta je definováno výchozí hodnotou, která je uvedena v sekci kanálu konfiguračního souboru klienta IBM MQ (`mqclient.ini`).

VYPNUTO
Automatické opětovné připojení klienta je vypnuto.

QMGR
IBM MQ classes for JMS pokus o připojení ke správci front se stejným identifikátorem správce front jako správce front, k němuž byl připojen, pomocí jedné z následujících voleb:

- Vlastnost **CONNECTIONNAMELIST** a kanál, který je definován ve vlastnosti **CHANNEL** .
- CCDT definované ve vlastnosti **CCDTURL** .

ANY
IBM MQ classes for JMS se pokusí znovu připojit ke správci front se stejným názvem, který má použití vlastnosti **CONNECTIONNAMELIST** nebo **CCDTURL**.

Související informace
stanza CHANNELS konfiguračního souboru klienta

Použití automatického opětovného připojení klienta v prostředí Java EE
Adaptér prostředků produktu IBM MQ , který lze implementovat do prostředí produktu Java EE (Java Platform, Enterprise Edition) a poskytovatele systému zpráv produktu WebSphere Application Server IBM MQ , může pomocí produktu IBM MQ classes for JMS komunikovat se správci front produktu IBM MQ . Adaptér prostředků produktu IBM MQ a poskytovatel systému zpráv produktu WebSphere Application Server IBM MQ poskytují podporu pro automatické opětovné připojení klienta.

Volby, které jsou k dispozici pro automatické opětovné připojení klienta v prostředí Java EE , jsou:

- Specifikace aktivace
- Porty modulu listener WebSphere Application Server
- Podnikové aplikace JavaBeans a webové aplikace
- Aplikace spuštěné uvnitř kontejnerů klienta

Poznámka: Automatické opětovné připojení klienta se specifikacemi aktivace pomocí funkčnosti poskytované produktem IBM MQ classes for JMS není podporováno. Adaptér prostředků produktu IBM MQ poskytuje vlastní mechanizmus pro opětovné připojení specifikací aktivace, pokud se správce front, ke kterému se specifikace aktivace připojuje, stane nedostupným.

Tento mechanismus je řízen:

- Vlastnost adaptéru prostředků IBM MQ **reconnectionRetryCount**.
- Vlastnost adaptéru prostředků IBM MQ **reconnectionRetryInterval**.
- Vlastnost specifikace aktivace **connectionNameList**.

Další informace o těchto vlastnostech naleznete v tématu “Konfigurace pro vlastnosti objektu ResourceAdapter” na stránce 413.

Použití automatického opětovného připojení klienta v rámci metody `onMessage()` objektu typu message-driven bean nebo jakékoli jiné aplikace spuštěné v rámci prostředí Java Platform, Enterprise Edition není podporováno. Aplikace potřebuje implementovat svou vlastní logiku opětovného připojení, pokud se správce front, k jehož připojení se připojuje, stane nedostupným.

Podpora pro automatické opětovné připojení klienta v prostředí Java EE
V prostředích Java EE , jako je například WebSphere Application Server, adaptér prostředků IBM MQ a poskytovatel systému zpráv produktu WebSphere Application Server IBM MQ poskytují podporu

pro automatické opětovné připojení klienta. Avšak v některých případech se na tuto podporu vztahují omezení.

Adaptér prostředků produktu IBM MQ , který lze implementovat do prostředí produktu Java EE a poskytovatele systému zpráv produktu WebSphere Application Server IBM MQ , lze pomocí produktu IBM MQ classes for JMS komunikovat se správci front produktu IBM MQ .

Následující tabulka shrnuje podporu, kterou adaptér prostředků produktu IBM MQ a poskytovatel systému zpráv produktu WebSphere Application Server IBM MQ poskytují podporu pro automatické opětovné připojení klienta.

Tabulka 45. Souhrn automatického opětovného připojení klienta v prostředí Java EE

	Vlastnost CONNECTIONNAMELIST	**Vlastnost CCDTURL**	**Vlastnost CLIENTRECONNECTOPTIONS**	**Alternativní přístup k automatickému opětovnému připojení klienta**
Specifikace aktivace	Podporováno s omezeními	Podporováno s omezeními	Nepodporováno	Prostředí Java EE a specifikace aktivace poskytují vlastní mechanizmus opětovného připojení.
Porty modulu listener WebSphere Application Server	Podporováno s omezeními	Podporováno s omezeními	Nepodporováno	WebSphere Application Server poskytuje svůj vlastní mechanizmus opětovného připojení
Podnikové aplikace JavaBeans a webové aplikace	Podporováno s omezeními	Podporováno s omezeními	Nepodporováno	Aplikace musí implementovat svou vlastní logiku opětovného připojení
Aplikace spuštěné uvnitř kontejnerů klienta	Podporováno	Podporováno	Podporováno	Nelze použít

Aplikace typu message-driven bean, které jsou instalovány v prostředí Java EE , jako např. IBM MQ classes for JMS, mohou používat specifikace aktivace ke zpracování zpráv na systému IBM MQ . Specifikace aktivace se používají k detekci zpráv, které přicházejí do systému IBM MQ a doručí je do objektů typu message-driven bean ke zpracování. Objekty typu message-driven bean mohou také vytvořit více připojení k systémům IBM MQ zevnitř jejich metody **onMessage()** . Další informace o tom, jak mohou tato připojení používat automatické opětovné připojení klienta, najdete v tématu Enterprise JavaBeans a webové aplikace.

Specifikace aktivace
Pro specifikace aktivace jsou podporovány vlastnosti **CONNECTIONNAMELIST** a **CCDTURL** s omezeními a vlastnost **CLIENTRECONNECTOPTIONS** není podporována.

Aplikace objektů typu message-driven bean (MDB), které jsou instalovány v prostředí produktu Java EE , jako například WebSphere Application Server, mohou používat specifikace aktivace ke zpracování zpráv na systému IBM MQ .

Specifikace aktivace se používají k detekci zpráv přicházejících do systému IBM MQ a poté je doručí do objektů MDB ke zpracování. Tento oddíl pojednává o tom, jak specifikace aktivace monitoruje systém IBM MQ .

Objekty MDB mohou také vytvářet další připojení k systémům IBM MQ z jejich metody onMessage() .

Podrobnosti o tom, jak mohou tato připojení používat automatické opětovné připojení klienta, lze nalézt v "Podnikové aplikace JavaBeans a webové aplikace" na stránce 265.

CONNECTIONNAMELIST vlastnost

Při spouštění se specifikace aktivace pokusí o připojení ke správci front pomocí:

- Jeden určený ve vlastnosti **QMANAGER** .
- Kanál uvedený ve vlastnosti **CHANNEL**
- Název hostitele a informace o portu z první položky v produktu **CONNECTIONNAMELIST**

Pokud se specifikace aktivace nemůže připojit ke správci front pomocí první položky v seznamu, přesune se specifikace aktivace na druhou položku atd. až do okamžiku, kdy bylo vytvořeno připojení ke správci front, nebo k dosažení konce seznamu.

Pokud se specifikace aktivace nemůže připojit k zadanému správci front pomocí některého z položek v produktu **CONNECTIONNAMELIST**, specifikace aktivace se zastaví a musí být znovu spuštěna.

Jakmile je specifikace aktivace spuštěna, specifikace aktivace získá zprávy ze systému IBM MQ a doručí zprávy do objektu MDB ke zpracování.

Pokud správce front selže při zpracování zprávy, prostředí produktu Java EE zjistí selhání a pokusí se znovu připojit specifikaci aktivace.

Specifikace aktivace používá informace ve vlastnosti **CONNECTIONNAMELIST** jako dříve, pokud specifikace aktivace provádí opakované pokusy o připojení.

Pokud se specifikace aktivace pokusí o všechny položky v produktu **CONNECTIONNAMELIST** a stále se nemůže připojit ke správci front, bude specifikace aktivace čekat po dobu určenou vlastností adaptéru prostředků produktu IBM MQ **reconnectionRetryInterval** před dalším pokusem.

Vlastnost adaptéru prostředku IBM MQ **reconnectionRetryCount** definuje počet následných pokusů o opětovné připojení, které se provedou před zastavením specifikace aktivace, a vyžaduje ruční restart.

Jakmile se specifikace aktivace znovu připojí k systému IBM MQ , prostředí Java EE provede všechny potřebné transakční vyčištění a pokračuje doručením zpráv do objektů MDB ke zpracování.

Aby bylo čištění transakcí správně fungovat, musí být prostředí produktu Java EE schopné přistupovat k protokolům pro správce front, který selhal.

Pokud jsou specifikace aktivace používány s transakčními objekty MDB, které se podílejí na transakcích XA, a jsou připojovány ke správci front s více instancemi, musí produkt **CONNECTIONNAMELIST** obsahovat položku pro aktivní i pro instanci správce front v pohotovostním režimu.

To znamená, že prostředí produktu Java EE může přistupovat k protokolům správců front v případě, že prostředí potřebuje provést zotavení transakce, bez ohledu na to, který správce front se prostředí znovu připojuje k následujícím selháním.

Pokud jsou transakční objekty MDB používány se samostatnými správci front, musí vlastnost **CONNECTIONNAMELIST** obsahovat jednu položku, aby se zajistilo, že se specifikace aktivace vždy znovu připojí ke stejnému správci front spuštěnému ve stejném systému, který následuje po selhání.

CCDTURL vlastnost

Při spouštění se specifikace aktivace pokusí o připojení ke správci front určenému ve vlastnosti **QMANAGER** pomocí první položky v tabulce definic kanálů klienta (CCDT).

Pokud se specifikace aktivace nemůže připojit ke správci front pomocí první položky v tabulce, přesune se specifikace aktivace na druhou položku atd. až do okamžiku, kdy bylo vytvořeno připojení ke správci front, nebo dokud není dosaženo konce tabulky.

Pokud se specifikace aktivace nemůže připojit k určenému správci front s použitím některého z položek v tabulce CCDT, specifikace aktivace se zastaví a musí být znovu spuštěna.

Jakmile je specifikace aktivace spuštěna, specifikace aktivace získá zprávy ze systému IBM MQ a doručí zprávy do objektu MDB ke zpracování.

Pokud správce front selže při zpracování zprávy, prostředí produktu Java EE zjistí selhání a pokusí se znovu připojit specifikaci aktivace.

Specifikace aktivace používá informace v rámci vlastnosti CCDT jako dříve, když specifikace aktivace provádí pokusy o opětovné připojení.

Pokud se specifikace aktivace pokusí o všechny položky v tabulce CCDT a stále se nemůže připojit ke správci front, bude specifikace aktivace čekat po dobu určenou vlastností adaptéru prostředků produktu IBM MQ **reconnectionRetryInterval** před dalším pokusem o opakování.

Vlastnost adaptéru prostředku IBM MQ **reconnectionRetryCount** definuje počet následných pokusů o opětovné připojení, které se provedou před zastavením specifikace aktivace, a vyžaduje ruční restart.

Jakmile se specifikace aktivace znovu připojí k systému IBM MQ , prostředí Java EE provede všechny potřebné transakční vyčištění a pokračuje doručením zpráv do objektů MDB ke zpracování.

Aby bylo čištění transakcí správně fungovat, musí být prostředí produktu Java EE schopné přistupovat k protokolům pro správce front, který selhal.

Pokud jsou specifikace aktivace používány s transakčními objekty MDB, které se účastní transakcí XA, a připojují se ke správci front s více instancemi, tabulka CCDT musí obsahovat položku pro aktivní i pro instanci správce front v pohotovostním režimu.

To znamená, že prostředí produktu Java EE může přistupovat k protokolům správců front v případě, že prostředí potřebuje provést zotavení transakce, bez ohledu na to, který správce front se prostředí znovu připojuje k následujícím selháním.

Pokud jsou transakční objekty MDB používány se samostatnými správci front, musí tabulka CCDT obsahovat jednu položku, aby se zajistilo, že se specifikace aktivace vždy znovu připojí ke stejnému správci front spuštěnému ve stejném systému, který následuje po selhání.

Ujistěte se, že jste nastavili výchozí hodnotu parametru *PREFERRED* pro vlastnost **AFFINITY** na CCDT, která se používá se specifikacemi aktivace, aby byla připojení vytvořena ke stejnému aktivnímu správci front.

CLIENTRECONNECTOPTIONS

Specifikace aktivace poskytují své vlastní funkce opětného připojení. Poskytnutá funkce umožňuje specifikacím automaticky znovu navázat spojení se systémem IBM MQ , pokud se správce front, k němuž byly připojeny, nezdaří.

Z tohoto důvodu není funkce automatického opětného připojení klienta poskytovaná serverem IBM MQ classes for JMS podporována.

Musíte nastavit vlastnost **CLIENTRECONNECTOPTIONS** na hodnotu *DISABLED* pro všechny specifikace aktivace, které se používají v produktu Java EE.

Porty modulu listener WebSphere Application Server
Aplikace objektů typu message-driven bean (MDB), které jsou instalovány v produktu WebSphere Application Server , mohou také používat porty modulu listener ke zpracování zpráv na systému IBM MQ .

Porty modulu listener se používají k detekci zpráv přicházejících do systému IBM MQ a poté je doručí do objektů MDB ke zpracování. Toto téma vysvětluje, jak port modulu listener monitoruje systém IBM MQ .

Objekty MDB mohou také vytvářet další připojení k systémům IBM MQ z jejich metody `onMessage()` .

Další informace o tom, jak tato připojení mohou používat automatické opětovné připojení klienta, viz “Podnikové aplikace JavaBeans a webové aplikace” na stránce 265 .

Pro porty modulu listener produktu WebSphere Application Server :

- **CONNECTIONNAMELIST** a **CCDTURL** jsou podporovány s omezeními
- **CLIENTRECONNECTOPTIONS** není podporováno

CONNECTIONNAMELIST

Porty modulu listener využívají fondy připojení prostoru JMS při připojování k produktu IBM MQ, a proto jsou předmětem důsledků použití fondů připojení. Další informace viz “Specifikace aktivace” na stránce 261.

Nejsou-li k dispozici žádná volná připojení a z této továrny připojení ještě nebyl vytvořen maximální počet připojení, použije se produkt **CONNECTIONNAMELIST** k pokusu o vytvoření nového připojení k produktu IBM MQ.

Nejsou-li všechny systémy IBM MQ v produktu **CONNECTIONNAMELIST** přístupné, port modulu listener se zastaví.

Port modulu listener poté čeká na časový úsek určený přizpůsobenou vlastností služby listener pro zprávy **RECOVERY.RETRY.INTERVAL** a pokusí se znovu znovu navázat spojení.

Tento pokus o opětovné připojení se pokusí zkontrolovat, zda ve fondu připojení nejsou nějaká volná připojení, pouze v případě, že byla vrácena mezi pokusy o připojení. Pokud jeden z nich není k dispozici, port modulu listener použije **CONNECTIONNAMELIST** jako dříve.

Jakmile se port modulu listener znovu připojí k systému IBM MQ , prostředí Java EE provede veškeré potřebné transakční vyčištění a pak bude pokračovat doručením zpráv do objektů MDB pro zpracování.

Aby bylo čištění transakcí správně fungovat, musí být prostředí produktu Java EE schopné přistupovat k protokolům pro správce front, který selhal.

Pokud jsou porty modulu listener používány s transakčními objekty MDB, které se podílejí na transakcích XA, a připojují se k **správci front s více instancemi**, musí produkt **CONNECTIONNAMELIST** obsahovat položku pro aktivní i pro instanci správce front v pohotovostním režimu.

To znamená, že prostředí produktu Java EE může přistupovat k protokolům správců front v případě, že prostředí potřebuje provést zotavení transakce, bez ohledu na to, který správce front se prostředí znovu připojuje k následujícím selháním.

Pokud jsou transakční objekty MDB používány se samostatnými správci front, musí vlastnost **CONNECTIONNAMELIST** obsahovat jednu položku, aby se zajistilo, že se specifikace aktivace vždy znovu připojí ke stejnému správci front spuštěnému ve stejném systému, který následuje po selhání.

CCDTURL

Při spouštění se port modulu listener pokusí o připojení ke správci front určenému ve vlastnosti **QMANAGER** pomocí první položky v tabulce CCDT.

Pokud se port modulu listener nemůže připojit ke správci front pomocí první položky v tabulce, port modulu listener se přesune na druhou položku a tak dále, dokud nebude navázáno spojení se správcem front nebo dokud nebude dosaženo konce tabulky.

Pokud se port modulu listener nemůže připojit k zadanému správci front pomocí některého z položek v tabulce CCDT, zastaví se port modulu listener.

Port modulu listener poté čeká na časový úsek určený přizpůsobenou vlastností služby listener pro zprávy **RECOVERY.RETRY.INTERVAL** a pokusí se znovu znovu navázat spojení.

Tento pokus o opětovné připojení funguje tak, aby prošel všemi položkami v tabulce CCDT jako dříve.

Je-li port modulu listener spuštěn, získává zprávy ze systému IBM MQ a doručuje je do objektu MDB ke zpracování.

Pokud správce front selže při zpracování zprávy, prostředí produktu Java EE zjistí selhání a pokusí se znovu připojit k portu modulu listener. Port modulu listener používá informace v tabulce CCDT při provádění pokusů o opětovné připojení.

Pokud se port modulu listener pokouší o všechny položky v tabulce CCDT a stále se nemůže připojit ke správci front, bude port čekat po dobu určenou hodnotou vlastnosti **RECOVERY.RETRY.INTERVAL** , než se znovu pokusí o připojení.

Vlastnost služby listener pro zprávy **MAX.RECOVERY.RETRIES** definuje počet následných pokusů o opětovné připojení, které se provedou před zastavením portu modulu listener a vyžaduje ruční restart.

Jakmile se port modulu listener znovu připojí k systému IBM MQ , prostředí Java EE provede veškeré potřebné transakční vyčištění a pak bude pokračovat doručením zpráv do objektů MDB pro zpracování.

Aby bylo čištění transakcí správně fungovat, musí být prostředí produktu Java EE schopné přistupovat k protokolům pro správce front, který selhal.

Pokud jsou porty modulu listener používány s transakčními objekty MDB, které se účastní transakcí XA, a připojují se ke správci front s více instancemi, tabulka CCDT musí obsahovat položku pro aktivní i pro instanci správce front v pohotovostním režimu.

To znamená, že prostředí produktu Java EE může přistupovat k protokolům správců front v případě, že prostředí potřebuje provést zotavení transakce, bez ohledu na to, který správce front se prostředí znovu připojuje k následujícím selháním.

Pokud jsou transakční objekty MDB používány se samostatnými správci front, tabulka CCDT musí obsahovat jednu položku, aby se zajistilo, že se port modulu listener vždy znovu připojí ke stejnému správci front, který je spuštěn ve stejném systému po selhání.

Ujistěte se, že jste nastavili výchozí hodnotu parametru *PREFERRED* pro vlastnost **AFFINITY** na CCDT, která se používá spolu s porty modulu listener, aby byla připojení vytvořena ke stejnému aktivnímu správci front.

CLIENTRECONNECTOPTIONS

Porty modulu listener zajišťují vlastní funkčnost opětovného připojení. Poskytnutá funkce umožňuje portu modulu listener automaticky znovu navázat spojení se systémem IBM MQ , pokud se správce front, k němuž byly připojeny, nezdaří.

Z tohoto důvodu není funkce automatického opětného připojení klienta poskytovaná serverem IBM MQ classes for JMS podporována.

Musíte nastavit vlastnost **CLIENTRECONNECTOPTIONS** na hodnotu *DISABLED* pro všechny porty modulu listener, které se používají v produktu Java EE.

Podnikové aplikace JavaBeans a webové aplikace
Aplikace Enterprise JavaBean (EJB) a aplikace spuštěné ve webovém kontejneru, jako např. Servlety, používají továrnu připojení produktu JMS k vytvoření připojení ke správci front produktu IBM MQ .

Pro sady EJB a webové aplikace platí následující omezení:

- **CONNECTIONNAMELIST** a **CCDTURL** jsou podporovány s omezeními
- **CLIENTRECONNECTOPTIONS** není podporováno

CONNECTIONNAMELIST

Pokud prostředí Java EE poskytuje fond připojení pro připojení JMS, viz “Použití CONNECTIONNAMELIST nebo CCDT ve fondu připojení” na stránce 267 , kde najdete informace o tom, jak tato vlastnost ovlivňuje chování vlastnosti **CONNECTIONNAMELIST** .

Pokud prostředí Java EE neposkytuje fond připojení JMS . aplikace používá vlastnost **CONNECTIONNAMELIST** stejným způsobem jako aplikace produktu Java SE .

Pokud jsou aplikace používány s transakčními objekty MDB, které se účastní transakcí XA, a které se připojují ke správci front s více instancemi, musí produkt **CONNECTIONNAMELIST** obsahovat položku pro aktivní i pro instanci správce front v pohotovostním režimu.

To znamená, že prostředí produktu Java EE může přistupovat k protokolům správců front v případě, že prostředí potřebuje provést zotavení transakce, bez ohledu na to, který správce front se prostředí znovu připojuje k následujícím selháním.

Pokud jsou aplikace používány se samostatnými správci front, musí vlastnost produktu **CONNECTIONNAMELIST** obsahovat jednu položku, aby bylo zajištěno, že se aplikace vždy znovu připojí ke stejnému správci front, který je spuštěn ve stejném systému, a to po selhání.

CCDTURL

Pokud prostředí Java EE poskytuje fond připojení pro připojení JMS , viz “Použití CONNECTIONNAMELIST nebo CCDT ve fondu připojení” na stránce 267 , kde získáte informace o tom, jak toto chování ovlivňuje chování vlastnosti **CCDTURL** .

Pokud prostředí Java EE neposkytuje fond připojení JMS . aplikace používá vlastnost **CCDTURL** stejným způsobem jako aplikace produktu Java SE .

Pokud jsou aplikace používány s transakčními objekty MDB, které se účastní transakcí XA, a připojují se ke správci front s více instancemi, tabulka CCDT musí obsahovat položku pro aktivní i pro instanci správce front v pohotovostním režimu.

To znamená, že prostředí produktu Java EE může přistupovat k protokolům správců front v případě, že prostředí potřebuje provést zotavení transakce, bez ohledu na to, který správce front se prostředí znovu připojuje k následujícím selháním.

Pokud jsou aplikace používány se samostatnými správci front, tabulka CCDT musí obsahovat jednu položku, aby se zajistilo, že se specifikace aktivace vždy znovu připojí ke stejnému správci front spuštěnému ve stejném systému, který následuje po selhání.

CLIENTRECONNECTOPTIONS

Musíte nastavit vlastnost **CLIENTRECONNECTOPTIONS** na hodnotu *DISABLED* pro všechny továrny připojení produktu JMS používané objekty EJB nebo aplikacemi, které se spouštějí ve webovém kontejneru.

Aplikace, které vyžadují automatické opětovné připojení k novému správci front, pokud správce front, kterého používáte, selže, je třeba implementovat vlastní logiku opětovného připojení. Další informace viz “Implementace logiky opětovného připojení v aplikaci Java EE” na stránce 268.

Scénáře: WebSphere Application Server s IBM MQ
Scénáře: Profil Liberty produktu WebSphere Application Server s produktem IBM MQ

Aplikace spuštěné uvnitř kontejnerů klienta
Některá prostředí Java EE , jako např. WebSphere Application Server, poskytují kontejner klienta, který lze použít ke spouštění aplikací produktu Java SE .

Aplikace spuštěné v rámci těchto prostředí používají továrnu připojení produktu JMS pro připojení ke správci front produktu IBM MQ .

Pro aplikace spuštěné uvnitř kontejnerů klienta:

- **CONNECTIONNAMELIST** a **CCDTURL** jsou plně podporovány
- Produkt **CLIENTRECONNECTOPTIONS** je plně podporován

CONNECTIONNAMELIST

Pokud prostředí Java EE poskytuje fond připojení pro připojení JMS, viz “Použití CONNECTIONNAMELIST nebo CCDT ve fondu připojení” na stránce 267 , kde najdete informace o tom, jak tato vlastnost ovlivňuje chování vlastnosti **CONNECTIONNAMELIST** .

Pokud prostředí Java EE neposkytuje fond připojení JMS . aplikace používá vlastnost **CONNECTIONNAMELIST** stejným způsobem jako aplikace produktu Java SE .

CCDTURL

Pokud prostředí Java EE poskytuje fond připojení pro připojení JMS , viz “Použití CONNECTIONNAMELIST nebo CCDT ve fondu připojení” na stránce 267 , kde získáte informace o tom, jak toto chování ovlivňuje chování vlastnosti **CCDTURL** .

Pokud prostředí Java EE neposkytuje fond připojení JMS . aplikace používá vlastnost **CCDTURL** stejným způsobem jako aplikace produktu Java SE .

Použití CONNECTIONNAMELIST nebo CCDT ve fondu připojení
Některá prostředí produktu Java EE , například WebSphere Application Server, poskytují fond připojení JMS . kontejner, který lze použít ke spuštění aplikací produktu Java SE .

Aplikace, které vytvářejí připojení s použitím továrny připojení definované v prostředí produktu Java EE , buď získají existující volné připojení z fondu připojení pro tuto továrnu připojení, nebo nové připojení, pokud ve fondu připojení není vhodný objekt.

To může mít důsledky, pokud byla továrna připojení nakonfigurována buď s definovanou vlastností **CONNECTIONNAMELIST** nebo **CCDTURL** .

Když se továrna připojení poprvé použije k vytvoření připojení, prostředí Java EE použije buď **CONNECTIONNAMELIST**. nebo **CCDTURL** , chcete-li vytvořit nové připojení k systému IBM MQ . Není-li toto připojení již vyžadováno, vrátí se do fondu připojení, kde je k dispozici připojení pro opětovné použití.

Pokud něco jiného vytvoří připojení z továrny na připojení, prostředí Java EE vrátí připojení z fondu připojení, spíše než pomocí vlastností **CONNECTIONNAMELIST** nebo **CCDTURL** vytvoří nové připojení.

Pokud se připojení používá, když selže instance správce front, připojení se vyřadí. Nicméně obsah fondu připojení nemusí být, což znamená, že fond může potenciálně stále obsahovat připojení ke správci front, který již není spuštěn.

V této situaci se při příští žádosti o vytvoření připojení z továrny připojení vrátí připojení k selhání správce front. Všechny pokusy o použití tohoto připojení selžou, protože správce front již není spuštěn, což způsobí vyřazení připojení.

Pouze v případě, že je fond připojení prázdný, bude prostředí produktu Java EE používat vlastnosti produktu **CONNECTIONNAMELIST** nebo **CCDTURL** k vytvoření nového připojení k produktu IBM MQ.

Vzhledem ke způsobu, jakým se **CONNECTIONNAMELIST** a CCDT používají k vytvoření JMS připojení, je možné mít také fond připojení, který obsahuje připojení k různým systémům IBM MQ .

Předpokládejme například, že továrna připojení byla konfigurována s vlastností **CONNECTIONNAMELIST** nastavenou na následující hodnotu:

```
CONNECTIONNAMELIST = hostname1(port1), hostname2(port2)
```

Předpokládejme, že při prvním pokusu aplikace o vytvoření připojení k samostatnému správci front z této továrny připojení není k dispozici správce front spuštěného v systému `hostname1(port1)` . To znamená, že aplikace skončí s připojením ke správci front spuštěnému v produktu `hostname2(port2)`.

Nyní se zobrazí jiná aplikace a vytvoří se připojení produktu JMS ze stejné továrny připojení. Správce front v produktu `hostname1(port1)` je nyní k dispozici, takže je vytvořeno nové připojení produktu JMS k tomuto systému IBM MQ a vrací se do aplikace.

Po dokončení obou aplikací zavřou produkt JMS Connections, což způsobí, že se připojení vrátí do fondu připojení.

Výsledkem je, že společná oblast připojení pro naši továrnu připojení nyní obsahuje dvě připojení JMS :

- Jedno připojení ke správci front spuštěnému v systému `hostname1(port1)`
- Jedno připojení ke správci front spuštěnému v systému `hostname2(port2)`

To může vést k problémům souvisejícím s obnovou transakcí. Pokud systém Java EE potřebuje odvolat transakci, musí být schopen se připojit ke správci front, který má přístup k protokolům transakcí.

Implementace logiky opětovného připojení v aplikaci Java EE
Objekty Enterprise JavaBeans a webové aplikace, které se chtějí automaticky znovu připojit, pokud správce front selže, je třeba implementovat vlastní logiku opětovného připojení.

Následující volby poskytují více informací o tom, jak toho lze dosáhnout:

Povolit selhání aplikace

Tento přístup nevyžaduje žádné změny aplikace, ale vyžaduje administrativní rekonfiguraci definice továrny připojení, aby obsahovala vlastnost **CONNECTIONNAMELIST** . Tento přístup však vyžaduje, aby původce volání byl schopen řádně zpracovat selhání. Všimněte si, že toto je také požadováno pro selhání, jako je MQRC_Q_FULL, která nesouvisí se selháním připojení.

Ukázkový kód pro tento proces:

```
public class SimpleServlet extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
       throws ServletException, IOException {
          try {
 // get connection factory/ queue
 InitialContext ic = new InitialContext();
 ConnectionFactory cf =
           (ConnectionFactory)ic.lookup("java:comp/env/jms/WMQCF");
 Queue q = (Queue) ic.lookup("java:comp/env/jms/WMQQueue");

 // send a message
 Connection c = cf.createConnection();
 Session s = c.createSession(false, Session.AUTO_ACKNOWLEDGE);
 MessageProducer p = s.createProducer(q);
 Message m = s.createTextMessage();
 p.send(m);

 // done, release the connection
 c.close();
 }
 catch (JMSException je) {
 // process exception
   }
 }
}
```

Předchozí kód předpokládá, že továrna připojení, kterou tento servlet používá, má definovanou vlastnost **CONNECTIONNAMELIST** .

Při prvním zpracování servletu je vytvořeno nové připojení s použitím vlastnosti **CONNECTIONNAMELIST** za předpokladu, že nejsou k dispozici žádná společná připojení z jiných aplikací připojujících se ke stejnému správci front.

Je-li připojení uvolněno po volání `close()` , je toto připojení vráceno do fondu a znovu použito při příštím spuštění servletu-bez odkazu na **CONNECTIONNAMELIST** -dokud nedojde k selhání připojení, ve kterém se vygeneruje událost CONNECTION_ERROR_OCCURRED. Tato událost vyzve fond, aby zničil neúspěšné připojení.

Při dalším spuštění aplikace nejsou k dispozici žádné připojení ve fondu a produkt **CONNECTIONNAMELIST** se používá pro připojení k prvnímu dostupnému správci front. Pokud došlo k překonání selhání správce front (například selhání nebylo přechodným selháním sítě), servlet se připojí k instanci zálohy, jakmile je k dispozici.

Jsou-li v aplikaci zahrnuty jiné prostředky, jako jsou databáze, může být vhodné označit, že by aplikační server měl transakci odvolat.

Zpracování opětovného připojení v rámci aplikace

Pokud původce volání nemůže ze servletu zpracovat selhání, je třeba v rámci aplikace zacházet s opětovným připojením. Jak je uvedeno v následujícím příkladu, pro zpracování opětovného připojení v rámci aplikace je vyžadována aplikace k vyžádání nového připojení tak, aby mohla být v mezipaměti továrny připojení, kterou vyhledal z produktu JNDI , a obsluhovali `JMSException` , jako například `JMSCMQ0001: VoláníWebSphere MQ selhalo s kódem compcode '2' ('MQCC_FAILED ')' 2009 '('MQRC_CONNECTION_BROKEN')`.

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {

  // get connection factory/ queue
  InitialContext ic = new InitialContext();
  ConnectionFactory cf = (ConnectionFactory)
              ic.lookup("java:comp/env/jms/WMQCF");
  Destination destination = (Destination) ic.lookup("java:comp/env/jms/WMQQueue");

  setupResources();

  // loop sending messages
  while (!sendComplete) {
    try {
      // create the next message to send
      msg.setText("message sent at "+new Date());
      // and send it
      producer.send(msg);
    }
    catch (JMSException je) {
        // drive reconnection
        setupResources();
    }
  }
```

V následujícím příkladu produkt `setupResources()` vytvoří objekty JMS a zahrnuje funkci spánku a smyčku opakování ke zpracování neokamžitých opěvovacích připojení. V praxi tato metoda zabraňuje mnoha pokusům o opakované připojení. Všimněte si, že podmínky ukončení byly vynechány z příkladu pro srozumitelnost.

```
 private void setupResources() {

    boolean connected = false;
    while (!connected) {
      try {
        connection = cf.createConnection(); // cf cached from JNDI lookup
        session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
        msg = session.createTextMessage();
        producer = session.createProducer(destination); // destination cached from JNDI lookup
        // no exception? then we connected ok
        connected = true;
      }
      catch (JMSException je) {
        // sleep and then have another attempt
        try {Thread.sleep(30*1000);} catch (InterruptedException ie) {}
      }
    }
```

Pokud aplikace spravuje opětovné připojení, je důležité, aby aplikace uvolňuje veškerá připojení k jiným prostředkům, ať už jsou tyto prostředky jiné správce front produktu IBM MQ nebo jiné back-endové služby, jako např. databáze. Tato připojení je třeba znovu vytvořit, pokud je opětovné připojení k nové instanci správce front IBM MQ dokončeno. Pokud neobnovujete připojení, prostředky aplikačního serveru jsou během pokusu o opětovné připojení zbytečně zadrženy a mohly by vypršet v době, kdy se znovu používají.

Použití správce WorkManager

Pro dlouho trvající aplikace (například dávkové zpracování), kde je doba zpracování delší než několik desítek sekund, lze použít správce WebSphere Application Server WorkManager . Následuje příklad fragmentu kódu pro WebSphere Application Server :

```
public class BatchSenderServlet extends HttpServlet  {

  private WorkManager workManager = null;
  private MessageSender sender; // background sender WorkImpl

  public void init() throws ServletException {
    InitialContext ctx = new InitialContext();
    workManager = (WorkManager)ctx.lookup(java:comp/env/wm/default);
    sender = new MessageSender(5000);
    workManager.startWork(sender);
  }

  public void destroy() {
    sender.halt();
  }

  public void doGet(HttpServletRequest req, HttpServletResponse res)
                              throws ServletException, IOException {
    res.setContentType("text/plain");
    PrintWriter out = res.getWriter();
    if (sender.isRunning()) {
      out.println(sender.getStatus());
    }
}
```

kde `web.xml` obsahuje:

```
<resource-ref>
      <description>WorkManager</description>
      <res-ref-name>wm/default</res-ref-name>
      <res-type>com.ibm.websphere.asynchbeans.WorkManager</res-type>
      <res-auth>Container</res-auth>
      <res-sharing-scope>Shareable</res-sharing-scope>
   </resource-ref>
```

a dávka se nyní implementuje přes pracovní rozhraní:

```
import com.ibm.websphere.asynchbeans.Work;

public class MessageSender implements Work {

  public MessageSender(int messages) {numberOfMessages = messages;}

  public void run() {
    // get connection factory/ queue
    InitialContext ic = new InitialContext();
    ConnectionFactory cf = (ConnectionFactory)
                ic.lookup("java:comp/env/jms/WMQCF");
    Destination destination = (Destination) ic.lookup("jms/WMQQueue");

    setupResources();

    // loop sending messages
    while (!sendComplete) {
      try {
        // create the next message to send
        msg.setText("message sent at "+new Date());
        // and send it
        producer.send(msg);
        // are we finished?
        if (sendCount == numberOfMessages) {sendComplete = true);
      }
      catch (JMSException je) {
          // drive reconnection
          setupResources();
      }
  }

  public boolean isRunning() {return !sendComplete;}
```

```
public void release() {sendComplete = true;}
```

Pokud zpracování dávkového zpracování trvá dlouho, například velké zprávy, pomalá síť nebo rozsáhlý přístup k databázi (zvláště když spojení s pomalým selháním) pak server spustí výstup zablokovaných varování podprocesů, podobně jako v následujícím příkladu:

WSVR0605W: Podproces "`WorkManager.DefaultWorkManager` : 0" (00000035) byl aktivní po 694061 milisekund a může být zablokovaný. Celkem je na serveru jedno nebo více vláken, které mohou být zablokované.

Tato varování lze minimalizovat zmenšením velikosti dávky nebo zvýšením časového limitu zablokovaného podprocesu. Je však obecně výhodnější, pokud toto zpracování implementujete v EJB (pro odeslání dávky) nebo ve zpracování objektu typu message driven bean (pro spotřebu nebo spotřebu a příjem).

Všimněte si, že opětovné připojení spravované aplikací neposkytuje obecné řešení pro zpracování běhových chyb, a aplikace musí stále zpracovat chyby, které nesouvisí s poruchou připojení.

Například při pokusu o vložení zprávy do fronty, která je plná (2053 MQRC_Q_FULL), nebo pokus o připojení ke správci front pomocí pověření zabezpečení, která nejsou platná (2035 MQRC_NOT_AUTHORIZED).

Aplikace musí také zpracovat chyby 2059 MQRC_Q_MGR_NOT_AVAILABLE, když nejsou žádné instance okamžitě k dispozici, když probíhá překonání selhání. Toho lze dosáhnout tím, že aplikace hlásí výjimky JMS tak, jak se vyskytují, namísto toho, aby se nebezobslužně pokoušela znovu navázat spojení.

Fondy objektů produktu IBM MQ classes for JMS
Použití fondu připojení mimo produkt Java EE pomáhá snížit celkové zatížení, například z některých samostatných aplikací za použití rámců nebo za účelem implementace do cloudových prostředí, a také z většího počtu klientských připojení do produktu `QueueManagers` , což vede ke zvýšení konsolidace serverů a správců front.

V rámci programovacího modelu Java EE je zde dobře definovaný životní cyklus různých objektů, které se používají. Objekty MDB (Message-driven bean) jsou nejomezenější, zatímco servlety poskytují více volného prostoru. Možnosti sdružování, které jsou k dispozici v rámci serverů Java EE , jsou proto používány pro různé použité programovací modely.

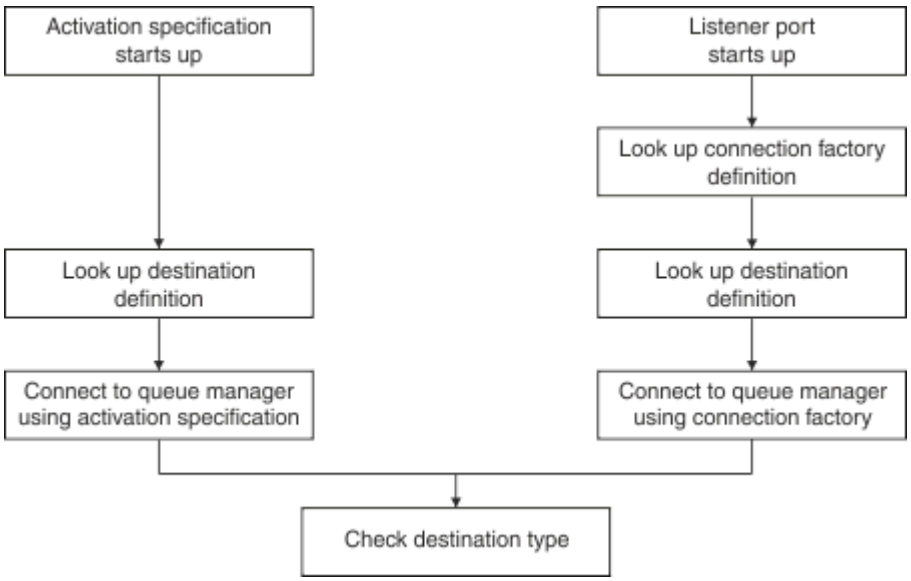
S produktem Java SE (nebo s jiným rámcem, jako je například Spring) jsou programovací modely extrémně flexibilní. Proto se jedna strategie sdružování nesluší všem. Měli byste uvažovat o tom, zda dojde k vytvoření rámce, který by mohl například použít sdružování do fondů, například Spring.

Strategie sdružování pro použití závisí na prostředí, v němž je spuštěna vaše aplikace.

Sdružování objektů do prostředí produktu Java EE
Aplikační servery produktu Java EE poskytují funkce sdružování připojení, které lze použít aplikacemi objektů bean řízenými zprávami, objekty Enterprise Java Bean a servlety.

Produkt WebSphere Application Server udržuje fond připojení k poskytovateli JMS za účelem zvýšení výkonu. Když aplikace vytvoří připojení k produktu JMS , aplikační server určí, zda již ve fondu volných připojení již existuje připojení. Je-li tomu tak, je připojení vráceno do aplikace; v opačném případě se vytvoří nové připojení.

Produkt Obrázek 47 na stránce 272 zobrazuje jak specifikace aktivace, tak i porty modulu listener zřídí připojení produktu JMS a používají toto připojení k monitorování cíle zpráv v normálním režimu.



Obrázek 47. Normální režim

Při použití poskytovatele systému zpráv produktu IBM MQ mohou aplikace, které provádějí odchozí systém zpráv (jako např. objekty Enterprise Java Bean a servlety) a komponentu portu modulu listener objektu typu message-driven bean, používat tyto fondy připojení.

Specifikaci aktivace poskytovatele systému zpráv produktu IBM MQ využívají funkce sdružování připojení poskytované adaptérem prostředků produktu IBM MQ . Další informace naleznete v tématu Konfigurace vlastností pro adaptér prostředků produktu WebSphere MQ .

Produkt “Příklady použití fondu připojení” na stránce 275 vysvětluje, jak aplikace, které provádějí odchozí systém zpráv, a porty modulu listener používají volný fond při vytváření připojení JMS .

Část “Volné podprocesy údržby fondu připojení” na stránce 278 vysvětluje, co se stane s těmito připojeními, když je aplikace nebo port modulu listener dokončen s připojeními.

Část “Příklady podprocesů údržby fondu” na stránce 279 vysvětluje, jak je volný fond připojení vyčištěn, aby nedošlo k zastarání připojení produktu JMS .

WebSphere Application Server má omezení počtu připojení, které lze vytvořit z továrny, určené vlastností *maximum connections* továrny připojení. Výchozí hodnota této vlastnosti je 10, což znamená, že v jednom okamžiku může být vytvořeno až 10 připojení vytvořených z továrny.

Každá továrna má přidružený volný fond připojení. Když se spustí aplikační server, jsou volné fondy připojení prázdné. Maximální počet připojení, která mohou existovat ve volném fondu pro továrnu, je také uveden ve vlastnosti Maximální počet připojení.

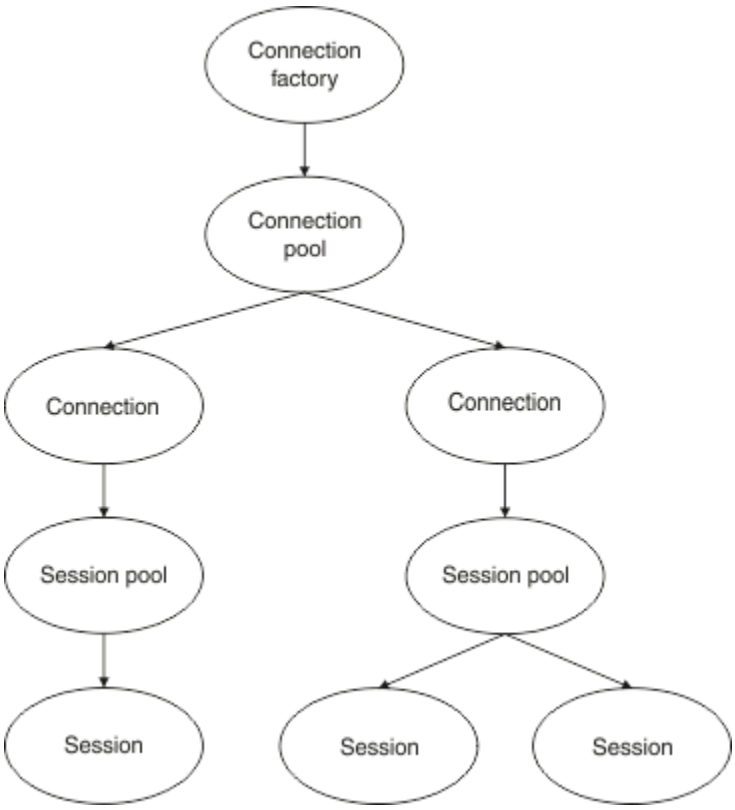
Tip: Pomocí produktu JMS 2.0lze továrnu připojení použít k vytvoření obou připojení a kontextů. Výsledkem je, že je možné mít k dispozici fond připojení přidružený k továrně připojení, která obsahuje kombinaci obou připojení a kontextů. Doporučuje se, aby továrna připojení byla použita pouze pro vytvoření připojení nebo vytvoření kontextů. Tím je zajištěno, že fond připojení pro tuto továrnu připojení bude obsahovat pouze objekty jednoho typu, což bude fond efektivnější.

Informace o tom, jak fond připojení funguje v produktu WebSphere Application Server, najdete v tématu Konfigurace fondů připojení pro připojení JMS. Informace o jiných aplikačních serverech najdete v příslušné dokumentaci aplikačního serveru.

Způsob použití fondu připojení

Ke každé faktorii připojení produktu JMS je přidružen fond připojení a fond připojení obsahuje nula nebo více připojení produktu JMS . Každé připojení JMS má přidružený fond relací JMS a každý fond relací JMS obsahuje žádný nebo více relací JMS .

Obrázek 48 na stránce 273 zobrazuje vztah mezi těmito objekty.



Obrázek 48. Fondy připojení a fondy relací

Pokud se spustí port modulu listener nebo pokud aplikace, která chce odchozí systém zpráv, používá továrnu k vytvoření připojení, volá port nebo aplikace jednu z následujících metod:

- **`connectionFactory.createConnection()`**
- **`ConnectionFactory.createConnection(String, String)`**
- **`QueueConnectionFactory.createQueueConnection()`**
- **`QueueConnectionFactory.createQueueConnection(String, String)`**
- **`TopicConnectionFactory.createTopicConnection()`**
- **`TopicConnectionFactory.createTopicConnection(String, String)`**

Správce připojení WebSphere Application Server se pokusí získat připojení z fondu volných prostředků pro tuto továrnu a vrátit jej do aplikace.

Pokud ve fondu nejsou žádná volná připojení a počet připojení vytvořených z této továrny nedosáhl limitu uvedeného ve vlastnosti *maximum connections* této továrny, správce Connection Manager vytvoří nové připojení pro aplikaci, která má být použita.

Pokud se však aplikace pokusí o vytvoření připojení, ale počet připojení vytvořených z této továrny se již rovná vlastnosti *maximum connections* továrny, aplikace čeká na to, až bude k dispozici připojení (aby bylo možné jej vrátit do volného fondu).

Čas, kdy aplikace čeká, je uveden ve vlastnosti *Časový limit připojení* fondu připojení, který má výchozí hodnotu 180 sekund. Je-li připojení ve fondu volných prostředků v rámci tohoto 180 sekund znovu uvedeno, správce Connection Manager ji ihned znovu odvede z fondu a předá jej aplikaci. Pokud však dojde k uplynutí časového limitu, dojde k vyvolání výjimky *ConnectionWaitTimeoutException* .

Když byla aplikace dokončena s připojením a uzavírá ji voláním:

- **`Connection.close()`**
- **`QueueConnection.close()`**
- **`TopicConnection.close()`**

připojení je skutečně otevřené a je vráceno do volného fondu, aby jej bylo možné znovu použít jinou aplikací. Proto můžete mít spojení otevřená mezi WebSphere Application Server a poskytovatelem JMS , i když na aplikačním serveru nejsou spuštěny žádné aplikace JMS .

Rozšířené vlastnosti fondu připojení
Existuje mnoho rozšířených vlastností, které lze použít k řízení chování fondů připojení produktu JMS .

Ochrana proti nárazem

"Jak aplikace, které provádějí odchozí zprávy, používají fond připojení" na stránce 277 popisuje použití metody `sendMessage()`, která zahrnuje `connectionFactory.createConnection()`.

Zvažte situaci, kdy máte 50 EJB, které vytvářejí připojení JMS ze stejné továrny připojení jako část jejich metody `ejbCreate()` .

Pokud jsou všechny tyto objekty bean vytvořeny ve stejnou dobu a ve fondu volných připojení továrny nejsou žádná připojení, aplikační server se pokusí vytvořit současně 50 JMS připojení ke stejnému poskytovateli JMS . Výsledkem je výrazné zatížení jak u WebSphere Application Server , tak u poskytovatele JMS .

Vlastnosti nárazové ochrany mohou této situaci zabránit tím, že omezí počet připojení produktu JMS , která lze vytvořit z továrny připojení v libovolném okamžiku, a ohromujících vytvoření dalších připojení.

Omezení počtu připojení produktu JMS v jednom okamžiku se dosáhne pomocí dvou vlastností:

- Prahová hodnota surge
- Interval nárazového vytváření.

Pokusí-li se aplikace EJB vytvořit připojení produktu JMS z továrny připojení, zkontroluje správce připojení, aby bylo vidět, kolik připojení se vytváří. Je-li tento počet menší nebo roven hodnotě vlastnosti `surge threshold` , bude správce připojení pokračovat v otevírání nových připojení.

Pokud však počet vytvářených připojení překročí vlastnost `surge threshold` , správce Connection Manager čeká po dobu určenou vlastností `surge creation interval` před vytvořením nového připojení a otevřením nového připojení.

Zastrčený připojení

Připojení JMS je považováno za `stuck`, pokud aplikace JMS používá toto připojení k odeslání požadavku poskytovateli produktu JMS a poskytovatel neodpoví v určitém časovém úseku.

Produkt WebSphere Application Server nabízí způsob, jak zjistit připojení produktu `stuck` JMS k použití této funkce, musíte nastavit tři vlastnosti:

- Časovač časovačů zablokování
- Čas zablokování
- Práh zablokování

Část "Příklady podprocesů údržby fondu" na stránce 279 vysvětluje, jak se podproces údržby fondu spouští pravidelně a kontroluje obsah volného fondu továrny připojení, hledá připojení, která buď nepoužitá po určitou dobu, nebo zda existují příliš dlouho trvající.

Chcete-li detekovat zablokovaná připojení, spravuje aplikační server zablokované připojení, které kontroluje stav všech aktivních připojení vytvořených z továrny připojení a zjišťuje, zda některý z nich čeká na odpověď od poskytovatele JMS .

Po spuštění zablokovaných podprocesů připojení je určena vlastnost `Stuck time timer` . Výchozí hodnota této vlastnosti je nula, což znamená, že detekce zablokování připojení se nikdy nespouští.

Pokud podproces najde odpověď čekající na odpověď, určí, jak dlouho bude čekat, a porovná tuto dobu s hodnotou vlastnosti `Stuck time` .

Pokud doba, po kterou má poskytovatel JMS odpovědět, překročí dobu určenou vlastností `Stuck time` , aplikační server označí připojení JMS jako zablokované.

Předpokládejme například, že továrna připojení jms/CF1 má nastavu vlastnost Stuck time timer na hodnotu 10 a vlastnost Stuck time je nastavena na hodnotu 15.

Podproces zablokované připojení se stane aktivním každých 10 sekund a kontroluje, zda každé připojení vytvořené z produktu jms/CF1 nečeká déle než 15 sekund na odpověď z produktu IBM MQ.

Předpokládejme, že objekt typu EJB vytvoří připojení produktu JMS k produktu IBM MQ pomocí produktu jms/CF1a pokusí se vytvořit relaci JMS s použitím tohoto připojení voláním příkazu Connection.createSession().

Avšak něco brání poskytovateli JMS v odezvě na požadavek. Možná je počítač zmrazen nebo proces spuštěný na poskytovateli JMS je zablokovaný a zabraňující zpracování jakékoli nové práce:

Deset sekund po volání EJB s názvem Connection.createSession()se časovač připojení stane aktivním a bude hledat aktivní připojení vytvořená z produktu jms/CF1.

Předpokládejme, že existuje pouze jedno aktivní připojení, například c1. První EJB čeká 10 sekund na odpověď na požadavek, který odeslal na c1, což je menší než hodnota Stuck time, takže časovač zablokovaných připojení bude toto připojení ignorovat a stane se neaktivním.

O 10 sekund později se zablokované vlákno připojení stane opět aktivním a prohledá aktivní spojení pro jms/CF1. Jako dříve předpokládejte, že existuje pouze jedno připojení, c1.

Nyní je to 20 sekund od prvního EJB nazvaného createSession()a objekt EJB stále čeká na odpověď. 20 sekund je delší než čas zadaný ve vlastnosti Stuck time , takže zablokovaná spojení vlákna připojení c1 označí jako zablokované.

Pokud o pět sekund později IBM MQ nakonec odpoví a umožní první sadě EJB vytvořit relaci JMS , je připojení opět používáno.

Aplikační server počítá počet připojení produktu JMS vytvořených z továrny připojení, která se zasekla. Když aplikace použije tuto továrnu připojení k vytvoření nového JMS připojení a ve fondu volných prostředků této továrny nejsou žádná volná připojení, správce připojení porovná počet zablokovaných připojení k hodnotě vlastnosti Stuck threshold .

Je-li počet zablokovaných připojení menší než hodnota nastavená pro vlastnost Stuck threshold, vytvoří správce připojení nové připojení a předá jej aplikaci.

Pokud je však počet zablokovaných připojení roven hodnotě vlastnosti Stuck threshold , aplikace získá výjimku prostředku.

Oblasti fondu

Produkt WebSphere Application Server poskytuje dvě vlastnosti, které umožňují rozdělení fondu volných připojení pro továrnu připojení:

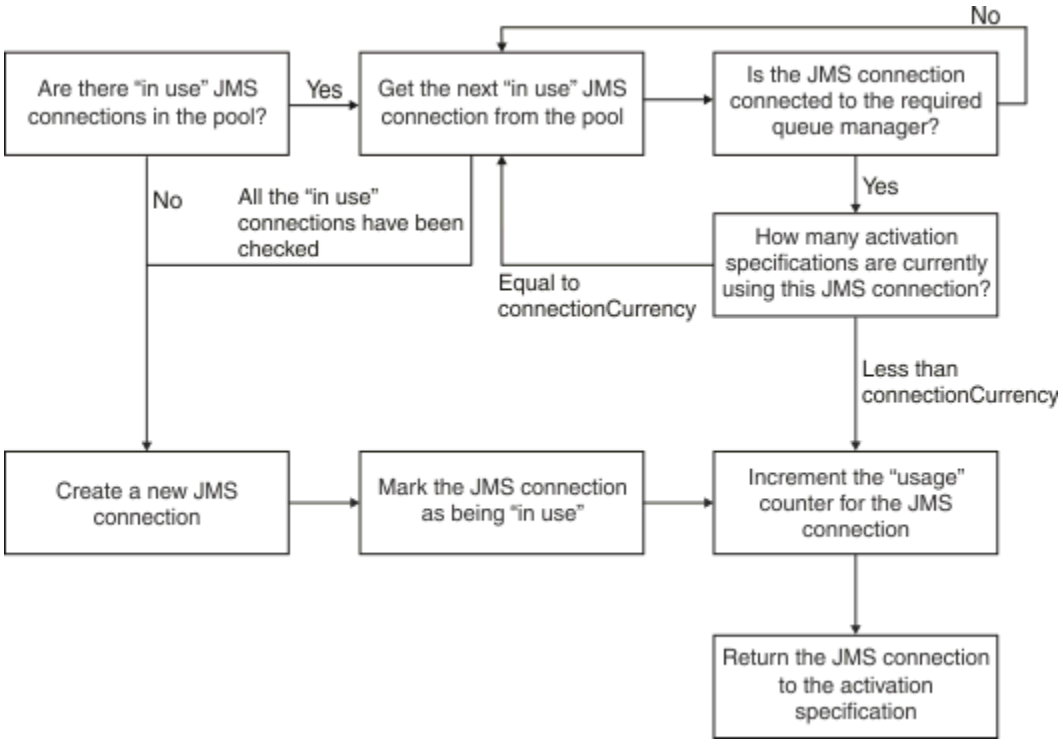
- Number of free pool partitions sděluje aplikačnímu serveru, kolik oblastí chcete rozdělit do fondu volných připojení.
- Free pool distribution table size určuje, jak jsou oblasti indexovány.

Ponechte tyto vlastnosti na výchozích hodnotách nula, pokud jste je nežádali o jejich změnu v Centru podpory produktu IBM .

Všimněte si, že produkt WebSphere Application Server má jednu další vlastnost rozšířeného fondu připojení s názvem Number of shared partitions. Tato vlastnost určuje počet oblastí používaných k ukládání sdílených připojení. Vzhledem k tomu, že připojení produktu JMS je vždy nesdílená, tato vlastnost se nepoužije.

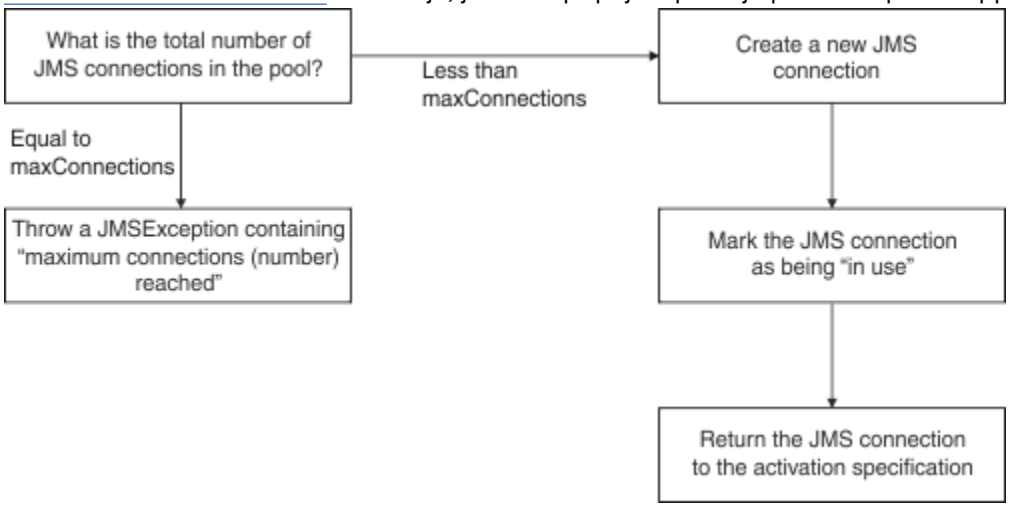
Příklady použití fondu připojení
Komponenta portu modulu listener řízeného zprávami a aplikace, které provádějí odchozí zasílání zpráv, používají fond připojení JMS .

Obrázek 49 na stránce 276 zobrazuje, jak fond připojení pracuje pro WebSphere Application Server 7.5 a 8.0.



Obrázek 49. WebSphere Application Server 7.5 a 8.0 -jak pracuje fond připojení

Obrázek 50 na stránce 276 zobrazuje, jak fond připojení pracuje pro WebSphere Application Server 8.5.



Obrázek 50. WebSphere Application Server 8.5 -jak pracuje fond připojení

Jak porty modulu listener MDB používají fond připojení

Předpokládejme, že máte objekt MDB implementovaný v systému WebSphere Application Server Network Deployment, který používá produkt IBM MQ jako poskytovatele JMS . Objekt MDB je implementován na portu modulu listener, který používá továrnu připojení s názvem, například `jms/CF1`, který má vlastnost *maximum connections* nastavenou na hodnotu 2, což znamená, že z této továrny může být vytvořeno pouze dvě připojení v libovolném okamžiku.

Při spuštění portu modulu listener se port pokusí vytvořit připojení k produktu IBM MQs použitím továrny připojení produktu`jms/CF1` .

Za tímto účelem port požaduje připojení od správce připojení. Protože se jedná o první použití továrny připojení produktu `jms/CF1` , neexistují žádná připojení ve fondu volného připojení produktu `jms/CF1` ,

takže správce připojení vytvoří nový volaný server, například c1. Všimněte si, že toto připojení existuje pro celý život portu modulu listener.

Nyní zvažte situaci, kdy zastavíte port modulu listener pomocí administrativní konzoly serveru WebSphere Application Server . V takovém případě správce připojení převezme spojení a vloží jej zpět do volného fondu. Připojení k produktu IBM MQ však zůstává otevřené.

Pokud restartujete port modulu listener, port znovu požádá správce připojení o připojení ke správci front. Vzhledem k tomu, že nyní máte ve volném fondu připojení (c1), správce připojení toto připojení odnese z fondu a zpřístupní jej pro port modulu listener.

Nyní předpokládejme, že máte druhý objekt MDB implementovaný na aplikační server, a používá jiný port modulu listener.

Předpokládejme, že se pokusíte spustit třetí port modulu listener, který je také nakonfigurovaný tak, aby používal továrnu připojení produktu `jms/CF1` . Třetí port modulu listener požaduje připojení od správce připojení, který vypadá ve volném fondu pro `jms/CF1` a zjistí, že je prázdný. Pak zkontroluje, kolik připojení již bylo vytvořeno z továrny `jms/CF1` .

Vzhledem k tomu, že vlastnost maximálního počtu připojení pro produkt `jms/CF1` je nastavena na hodnotu 2 a vy jste již vytvořili dvě připojení z této továrny, bude správce připojení čekat 180 sekund (výchozí hodnota vlastnosti časového limitu připojení) pro připojení, které má být k dispozici.

Pokud však zastavíte první port modulu listener, jeho připojení c1 se vloží do volného fondu pro prostor `jms/CF1`. Správce připojení načte toto připojení a předá jej třetímu modulu listener.

Pokud se nyní pokusíte restartovat první modul listener, musí tento modul listener čekat na zastavení jednoho z ostatních portů modulu listener, než se bude moci restartovat první modul listener. Pokud se žádný z běžících portů modulu listener nezastaví během 180 sekund, obdrží první modul listener chybu `ConnectionWaitTimeoutException` a zastaví se.

Jak aplikace, které provádějí odchozí zprávy, používají fond připojení

U této volby předpokládejme, že existuje jediný objekt EJB s názvem, například EJB1, který je instalován na aplikačním serveru. Objekt typu bean implementuje metodu s názvem `sendMessage()` tím, že:

- Vytvoření připojení JMS k IBM MQ z továrny `jms/CF1`pomocí produktu `connectionFactory.createConnection()`.
- Vytvoření relace JMS z připojení.
- Vytvoření producenta zpráv z relace.
- Odesílá se zpráva.
- Zavírám výrobce.
- Zavírá se relace.
- Zavírá se spojení voláním `connection.close()`.

Předpokládejme, že volný fond pro továrnu `jms/CF1` je prázdný. Při prvním vyvolání objektu EJB se objekt typu bean pokusí vytvořit připojení k produktu IBM MQ z továrny `jms/CF1`. Vzhledem k tomu, že volný fond pro továrnu je prázdný, vytvoří správce připojení nové připojení a předá jej EJB1.

Těsně před tím, než metoda skončí, volá metoda `connection.close()`. Místo uzavření `c1`správce připojení převezme spojení a vkládá jej do volného fondu pro prostor `jms/CF1`.

Při příštím volání `sendMessage()` se metoda `connectionFactory.createConnection()` vrátí do aplikace `c1` .

Předpokládejme, že máte druhou instanci EJB spuštěné ve stejnou dobu jako první instance. Když obě instance volají `sendMessage()`, vytvoří se dvě připojení z továrny připojení `jms/CF1` .

Nyní předpokládejme, že je vytvořena třetí instance objektu typu bean. Když třetí objekt typu bean vyvolá `sendMessage()`, volá metoda `connectionFactory.createConnection()` , aby vytvořil spojení z `jms/CF1`.

Nyní však existují dvě připojení vytvořená z jms/CF1, která se rovná hodnotě maximálního počtu připojení pro tuto továrnu. Proto metoda createConnection() čeká 180 sekund (výchozí hodnota vlastnosti časového limitu připojení) pro připojení, které má být k dispozici.

Pokud však metoda sendMessage() pro první volání EJB connection.close() a skončí, připojení, které používal, c1, se vrátí do volného fondu připojení. Správce připojení vezme připojení zpět z volného fondu a předá jej třetímu EJB. Volání z tohoto objektu typu bean na connectionFactory.createConnection() se pak vrátí, což umožní dokončení metody sendMessage(.

Porty modulu listener MDB a objekty EJB používající stejný fond připojení

Dva předchozí příklady ukazují, jak porty modulu listener a objekty EJB mohou používat fond připojení v izolaci. Můžete však mít port modulu listener i EJB spuštěné uvnitř stejného aplikačního serveru a vytvářet připojení produktu JMS se stejnou faktorií připojení.

Měli byste zvážit důsledky této situace

Klíčovou věcí je pamatovat na to, že továrna připojení je sdílena mezi portem modulu listener a objektem EJB.

Předpokládejme například, že máte spuštěný modul listener a modul EJB. Oba používají továrnu připojení produktu jms/CF1 , což znamená, že byl dosažen limit připojení určený vlastností maximálního počtu připojení pro danou továrnu.

Pokud se pokusíte spustit buď jiný port modulu listener nebo jinou instanci objektu EJB, musíte počkat, až se vrátí připojení k fondu volných připojení pro produkt jms/CF1.

Volné podprocesy údržby fondu připojení
Přidružen ke každému fondu volného připojení je podproces údržby fondu, který monitoruje volný fond, aby se ujistil, že připojení jsou stále platná.

Pokud se podproces údržby fondu rozhodne, že je nutné zrušit připojení ve volném fondu, podproces fyzicky uzavře připojení produktu JMS k produktu IBM MQ.

Jak pracuje podproces údržby fondu

Chování podprocesu údržby fondu se určuje podle hodnoty čtyř vlastností fondu připojení:

Časový limit životnosti
Doba, po kterou zůstane připojení otevřené.

Minimální počet připojení
Minimální počet připojení, které správce připojení uchovává ve volném fondu pro továrnu připojení.

Interval spuštění
Jak často se podproces údržby fondu spouští.

Časový limit nečinnosti
Jak dlouho zůstane připojení ve volném fondu, než bude zavřeno.

Ve výchozím nastavení se spravovaný fond spouští každých 180 sekund, ačkoli tuto hodnotu lze změnit nastavením vlastnosti fondu připojení **Reap time** .

Podproces údržby se dívá na každé připojení ve fondu, kontroluje, jak dlouho byl ve fondu a kolik času uplynulo od doby, kdy byl vytvořen a naposledy použit.

Pokud připojení nebylo použito pro období delší, než je hodnota vlastnosti **Unused timeout** pro fond připojení, podproces údržby zkontroluje počet připojení, která jsou aktuálně ve volném fondu. Pokud je toto číslo:

- Větší než hodnota **Minimum connections**, správce připojení zavře připojení.
- Equals the value of **Minimum connections**, connection is not closed and remains in the free pool.

Výchozí hodnota vlastnosti **Minimum connections** je *1*, což znamená, že z důvodu výkonu se správce Connection Manager vždy pokusí zachovat alespoň jedno připojení ve volném fondu.

Vlastnost **Unused timeout** má výchozí hodnotu 1800 sekund. Je-li připojení ve volném fondu opět ve volném fondu a není znovu použito po dobu nejméně 1800 sekund, je toto připojení zavřeno, za předpokladu, že jej zavřou, ponechá se alespoň jedno připojení ve volném fondu.

Tento postup zabraňuje tomu, aby se nepoužívané připojení staly zastaralými. Chcete-li tuto funkci vypnout, nastavte vlastnost **Unused timeout** na nulu.

Je-li připojení ve volném fondu a uplynulá doba od jeho vytvoření je větší než hodnota vlastnosti **Aged timeout** pro fond připojení, pak je uzavřena bez ohledu na to, jak dlouho byla od jeho posledního použití použita.

Ve výchozím nastavení je vlastnost **Aged timeout** nastavena na nulu, což znamená, že podproces údržby tuto kontrolu nikdy neprovede. Připojení, která jsou delší než vlastnost **Aged timeout** , jsou zahozena bez ohledu na to, kolik připojení zůstane ve volném fondu. Všimněte si, že vlastnost **Minimum connections** nemá v této situaci žádný vliv.

Vypnutí podprocesu údržby fondu

Z předchozího popisu můžete vidět, že podproces údržby fondu provádí velké množství práce, je-li aktivní, zvláště pokud ve volném fondu továrny připojení existuje velký počet připojení.

Předpokládejme například, že existují tři továrny připojení JMS s vlastností **Maximum connections** nastavenou na hodnotu 10 pro každou továrnu. Každých 180 sekund se tři podprocesy údržby fondu stanou aktivními a skenují volné společné oblasti pro každou továrnu na připojení. Pokud mají volné fondy mnoho připojení, podprocesy údržby mají mnoho práce, což může výrazně ovlivnit výkon.

Můžete zakázat podproces údržby fondu pro jednotlivý fond volného připojení nastavením jeho vlastnosti **Reap time** na nulu.

Vypnutí podprocesu údržby znamená, že připojení nejsou nikdy zavřena, i když již **Unused timeout** uplynula. Avšak, připojení mohou být stále uzavřena, pokud **Aged timeout** prošel.

Pokud byla aplikace dokončena s připojením, zkontroluje správce připojení, jak dlouho již existuje připojení, a pokud je toto období delší než hodnota vlastnosti **Aged timeout** , správce Connection Manager toto připojení spíše zavře, než aby jej vracel do volného fondu.

Transakční důsledky vypršení časového limitu agregace

Jak je popsáno v předchozí sekci, vlastnost **Aged timeout** uvádí, jak dlouho zůstane připojení k poskytovateli JMS otevřené před tím, než správce připojení zavře.

Výchozí hodnota pro vlastnost **Aged timeout** je nula, což znamená, že připojení nebude nikdy zavřeno, protože je příliš staré. Vlastnost **Aged timeout** byste měli ponechat na této hodnotě, protože povolení **Aged timeout** může mít transakční dopady při použití JMS uvnitř EJB.

V JMSje jednotka transakce JMS *relace*, která je vytvořena z připojení JMS *connection*. Jedná se o JMS *relaci* , která je uvedena do transakcí, a ne JMS *připojení*.

Kvůli návrhu aplikačního serveru lze spojení JMS uzavřít, protože **Aged timeout** již uplynulo, i když jsou relace JMS vytvořené z tohoto připojení zapojeny do transakce.

Uzavření připojení produktu JMS způsobí, že všechny nevyřízené transakce v relacích JMS budou odvolány, jak je popsáno ve specifikaci JMS . Aplikační server si však není vědom toho, že relace JMS vytvořené z připojení již nejsou platné. Když se server pokusí použít relaci k potvrzení nebo odvolání transakce, dojde k `IllegalStateException` .

Důležité: Chcete-li použít produkt **Aged timeout** s připojeními produktu JMS z EJB, ujistěte se, že každá práce JMS je explicitně potvrzena na relaci JMS před tím, než metoda EJB, která provádí operace JMS , ukončí svou činnost.

Příklady podprocesů údržby fondu
Pomocí objektu EJB (Enterprise Java Bean) můžete pochopit, jak podproces údržby fondu pracuje. Všimněte si, že můžete také použít objekty MDB (Message Driven Beans) a porty modulu listener, protože vše, co potřebujete, je způsob, jak získat připojení ve volném fondu.

Další podrobnosti o metodě `sendMessage()` viz “Jak aplikace, které provádějí odchozí zprávy, používají fond připojení” na stránce 277 .

Konfigurovali jste továrnu na připojení s následujícími hodnotami:

- **`Reap time`** při výchozí hodnotě 180 sekund
- **`Aged timeout`** při výchozí hodnotě nula sekund
- **`Unused timeout`** nastaveno na 300 sekund

Po spuštění aplikačního serveru je vyvolána metoda `sendMessage()` .

Metoda vytvoří připojení s názvem, například `c1`za použití továrny `jms/CF1`, používá tuto továrnu k odeslání zprávy a pak zavolá příkaz `connection.close()`, který způsobí, že se `c1` umístí do volného fondu.

Po 180 sekundách se podproces údržby fondu spustí a podívá se na fond volného připojení `jms/CF1` . Ve fondu je volné připojení `c1` , takže podproces údržby se dívá na čas, kdy bylo připojení uvedeno zpět, a porovnává je s aktuálním časem.

Od doby, kdy bylo připojení uvedeno ve volném fondu, uplynulo 180 sekund, což je méně než hodnota vlastnosti **`Unused timeout`** pro `jms/CF1`. Podproces údržby proto ponechá připojení samotné.

O 180 sekund později se podproces údržby fondu spustí znovu. Podproces údržby najde připojení `c1`a zjistí, že připojení bylo ve fondu 360 sekund, což je delší než nastavená hodnota **`Unused timeout`** , takže správce připojení uzavře připojení.

Pokud nyní znovu spustíte metodu `sendMessage()` , vytvoří-li aplikace `connectionFactory.createConnection()` , správce připojení vytvoří nové připojení k produktu IBM MQ , protože volný fond připojení pro továrnu připojení je prázdný.

Předchozí příklad ukazuje, jak podproces údržby používá vlastnosti **`Reap time`** a **`Unused timeout`** k zabránění zablokovaným připojením, je-li vlastnost **`Aged timeout`** nastavena na nulu.

Jak vlastnost **`Aged timeout`** funguje?

V následujícím příkladu předpokládejme, že jste nastavili:

- Vlastnost **`Aged timeout`** na 300 sekund
- Vlastnost **`Unused timeout`** na nulu.

Vyvoláváte metodu `sendMessage()` a tato metoda se pokusí vytvořit připojení z továrny připojení `jms/CF1` .

Vzhledem k tomu, že volný fond pro tuto továrnu je prázdný, vytvoří správce připojení nové připojení `c1`a vrátí jej do aplikace. Když `sendMessage()` volá `connection.close()`, `c1` se umístí zpět do volného fondu připojení.

O 180 sekund později se podproces údržby fondu spustí. Vlákno najde `c1` ve fondu volného připojení a kontroluje, jak dlouho byl vytvořen. Připojení bylo vytvořeno 180 sekund, což je méně než **`Aged timeout`**, takže podproces údržby fondu jej ponechá samostatně a přejde zpět do režimu spánku.

O 60 sekund později je příkaz `sendMessage()` znovu volán. Tento čas, když metoda volá `connectionFactory.createConnection()`, zjistí správce připojení, že existuje připojení, `c1`, dostupné ve volném fondu pro `jms/CF1`. Správce připojení vezme produkt `c1` mimo volný fond a dá toto připojení k aplikaci.

Když se `sendMessage()` ukončí, vrátí se do volného fondu připojení. O 120 sekund později se podproces údržby fondu znovu probudí, prohledá obsah volného fondu pro `jms/CF1` a zjistí `c1`.

Ačkoli připojení bylo použito pouze před 120 sekundami, podproces údržby fondu zavře připojení, protože připojení existuje celkem 360 sekund, což je delší než hodnota 300 sekund, kterou jste nastavili pro vlastnost **`Aged timeout`** .

Jak vlastnost Minimální připojení ovlivňuje podproces údržby fondu

Použití příkladu “Jak porty modulu listener MDB používají fond připojení” na stránce 276 znovu předpokládá, že máte na aplikačním serveru implementovány dva objekty MDB, přičemž každý z nich používá jiný port modulu listener.

Každý port modulu listener je konfigurován tak, aby používal továrnu připojení produktu `jms/CF1` , kterou jste nakonfigurovali s použitím:

- Vlastnost **`Unused timeout`** byla nastavena na 120 sekund.
- Vlastnost **`Reap time`** nastavena na 180 sekund
- Vlastnost **`Minimum connections`** nastavena na 1

Předpokládejme, že první modul listener je zastaven a jeho připojení `c1` je zahrnuto do volného fondu. O 180 sekund později se podproces údržby fondu probudí, prohledá obsah volného fondu pro `jms/CF1`a zjistí, že `c1` je ve volném fondu déle, než hodnota vlastnosti **`Unused timeout`** pro továrnu připojení.

Před zavřením produktu `c1`se však podproces údržby fondu pokusí zjistit, kolik připojení zůstane ve fondu, pokud je toto připojení zahozena. Vzhledem k tomu, že produkt `c1` je jediné připojení ve fondu volných připojení, správce Connection Manager ji nezavře, protože tak učiní počet připojení, která zůstanou ve fondu volných prostředků, menší než hodnota nastavená pro produkt **`Minimum connections`**.

Nyní předpokládejme, že je druhý modul listener zastaven. Fond volných připojení nyní obsahuje dvě volná připojení- `c1` a `c2`.

O 180 sekund později se podproces údržby fondu spustí znovu. Po této době `c1` byl ve volném spojovacím fondu po dobu 360 sekund a `c2` po dobu 180 sekund.

Podproces údržby fondu kontroluje produkt `c1` a zjišťuje, že se nachází ve fondu déle, než je hodnota vlastnosti **`Unused timeout`** .

Tento podproces pak zjišťuje, kolik připojení je ve volném fondu, a porovnává je s hodnotou vlastnosti **`Minimum connections`** . Jelikož fond obsahuje dvě připojení a **`Minimum connections`** je nastaven na 1, správce připojení zavře `c1`.

Podproces údržby se nyní dívá na `c2`. To bylo také ve volném fondu připojení déle, než hodnota vlastnosti **`Unused timeout`** . Protože však uzavření `c2` ponechá volný fond připojení s méně než nastaveným počtem minimálního počtu připojení, správce Connection Manager ponechá produkt `c2` samostatně.

Připojení JMS a IBM MQ
Informace o použití IBM MQ jako poskytovatele JMS .

Použití přenosu vazeb

Pokud byla továrna připojení konfigurována pro použití přenosu vazeb, každé připojení produktu JMS naváže konverzaci (také známou jako **hconn**) s IBM MQ. Konverzace používá komunikaci mezi procesy (nebo sdílenou paměť) ke komunikaci se správcem front.

Použití přenosu klienta

Je-li továrna připojení poskytovatele systému zpráv produktu IBM MQ konfigurována pro použití transportu klienta, vytvoří každé připojení vytvořené z této továrny novou konverzaci (také známou jako**hconn**) k produktu IBM MQ.

V případě továren připojení, které se připojují ke správci front s použitím běžného režimu poskytovatele systému zpráv produktu IBM MQ , je možné, aby bylo vytvořeno více připojení produktu JMS vytvořených z továrny připojení ke sdílení připojení TCP/IP k produktu IBM MQ. Další informace viz Sdílení připojení TCP/IP v produktu IBM MQ classes for JMS.

Chcete-li určit maximální počet kanálů klienta používaných připojeními produktu JMS v jednom okamžiku, přidejte hodnotu vlastnosti *Maximální počet připojení* pro všechny faktorie připojení, které odkazují na stejného správce front.

Předpokládejme například, že máte dvě továrny připojení, `jms/CF1` a `jms/CF2` , které byly nakonfigurovány pro připojení ke stejnému správci front IBM MQ pomocí stejného kanálu IBM MQ .

Tyto továrny používají výchozí vlastnosti fondu připojení, což znamená, že *Maximální počet připojení* je nastaven na 10. Pokud jsou všechna připojení používána z obou položek jms/CF1 a jms/CF2 současně, bude mezi aplikačním serverem a produktem IBM MQprobíhat 20 konverzací.

Pokud se továrna připojení připojuje ke správci front s použitím běžného režimu poskytovatele systému zpráv produktu IBM MQ , pak maximální počet připojení TCP/IP, která mohou existovat mezi aplikačním serverem a správcem front pro tyto továrny připojení, je následující:

```
20/the value of SHARECNV for the IBM MQ channel
```

Je-li továrna připojení konfigurována pro připojení s použitím režimu migrace poskytovatele systému zpráv produktu IBM MQ , bude maximální počet připojení TCP/IP mezi aplikačním serverem a produktem IBM MQ pro tyto továrny připojení 20 (jeden pro každé připojení JMS ve fondech připojení pro tyto dvě továrny).

Související informace

použitíIBM MQ classes for JMS

Sdružování objektů do prostředí produktu Java SE

S produktem Java SE (nebo s jiným rámcem, jako je například Spring) jsou programovací modely extrémně flexibilní. Proto se jedna strategie sdružování nesluší všem. Měli byste zvážit, zda existuje rámec na místě, který by mohl dělat jakoukoli formu sdružování, například Spring.

Jinak by mohla logika aplikace provést tuto operaci. Zeptejte se sami sebe, jak složité je aplikace samotná? Je nejlepší porozumět aplikaci a požadavkům na připojení k systému zasílání zpráv. Aplikace jsou často napsány i v rámci svého vlastního kódu obálky kolem základního rozhraní API produktu JMS .

I když to může být velmi rozumný přístup a může skrývat složitost, stojí za to mít na paměti, že to může představovat problémy. Například generická metoda `getMessage()` , která se často volá, by neměla jen otevírat a zavírat spotřebitele.

Body, které byste měli zvážit:

- Jak dlouho bude aplikace potřebovat přístup k produktu IBM MQ? Celou dobu, nebo jen občas.
- Jak často budou odesílány zprávy? Čím méně často, tím více může být jedno připojení k IBM MQ sdílené.
- Výjimka přerušení spojení je obvykle známkou nutnosti opakovaného vytvoření připojení ve fondu. Co třeba:
  - Výjimka zabezpečení nebo hostitel není k dispozici
  - Úplná výjimka fronty
- Vyskytne-li se výjimka při přerušení spojení, co by se mělo stát s ostatními volnými připojeními ve fondu? Měly by být uzavřeny a znovu vytvořeny?
- Je-li například použit protokol TLS, jak dlouho chcete, aby zůstalo jedno připojení otevřené?
- Způsob, jakým se připojení do fondu identifikuje takovým způsobem, že administrátor správce front může připojení zjistit a sledovat jej zpět.

Měli byste vzít v úvahu všechny objekty JMS pro sdružování a objekty fondu, kdykoli je to možné provést. Mezi tyto objekty patří:

- JMS připojení
- Relace
- Kontexty
- Výrobci a spotřebitelé všech různých typů

Při použití transportu klienta budou při komunikaci se správcem front produktu IBM MQ používány sokety, připojení, relace a kontexty produktu JMS . Sdílením těchto objektů se úspory nacházejí na počtu příchozích připojení produktu IBM MQ (hConns) na správce front a snížení počtu instancí kanálu.

Použití přenosu vazeb na správce front odebere síťovou vrstvu zcela. Mnoho aplikací však používá přenos klienta k zajištění vyšší dostupnosti a vyrovnávání pracovní zátěže, konfigurace.

Výrobci JMS a spotřebitelé otevřou místa určení ve správci front. Pokud je otevřeno méně čísel front nebo témat a více částí aplikace tyto objekty používá, je možné je vzájemně sdílet.

Z pohledu produktu IBM MQ tento proces ukládá sekvenci operací MQOPEN a MQCLOSE.

Připojení, relace a kontexty

Tyto objekty všechny zapouzdřují obslužné rutiny připojení IBM MQ ke správci front a jsou generovány z `ConnectionFactory`. Do aplikace můžete přidat logiku pro omezení počtu připojení a dalších objektů vytvořených z jedné továrny připojení k určitému číslu.

V aplikaci můžete použít jednoduchou strukturu dat, která bude obsahovat vytvářená připojení. Kód aplikace, který potřebuje použít jednu z těchto struktur dat, může *zapůjčit* objekt, který se má použít.

Vezměte v úvahu následující faktory:

- Kdy mají být připojení odebrána z fondu? Obecně vytvořte na připojení modul listener pro výjimky. Je-li modul listener volán ke zpracování výjimky, měli byste znovu vytvořit připojení a všechny relace vytvořené z tohoto připojení.
- Je-li tabulka CCDT používána pro vyrovnávání pracovní zátěže, může připojení přejít k různým správcům front. To může být vhodné pro požadavky na sdružování.

Nezapomeňte, že specifikace JMS udává, že se jedná o chybu programování pro více podprocesů, které mají současně přistupovat k relaci nebo ke kontextu. Kód IBM MQ JMS se snaží být přísný při manipulaci s podprocesy. Do aplikace byste však měli přidat logiku, abyste se ujistili, že objekt nebo objekt kontextu je používán pouze jedním vláknem v daném okamžiku.

Výrobci a spotřebitelé

Každý producent a odběratel, který je vytvořen, otevře místo určení ve správci front. Pokud má být stejné místo určení použito pro celou řadu úloh, má smysl zachovat otevřené objekty spotřebitele nebo producenta. Zavřete objekt pouze, když se provede veškerá práce.

Ačkoli otevření a uzavření cíle jsou krátké operace, pokud jsou provedeny často, může čas, kdy se může čas přidat.

Rozsah těchto objektů je v rámci relace nebo kontextu, ze kterého jsou vytvářeni, proto je třeba je v rámci daného rozsahu platnosti. Obecně platí, že aplikace jsou napsány tak, aby to bylo zcela jednoduché dělat.

Monitorování

Jak budou aplikace monitorovat jejich fondy objektů? Odpověď na tuto otázku je do značné míry určována komplexností řešení, které má být prováděno do fondů.

Pokud považujete implementaci fondu Java EE za použití fondu, existuje velký počet voleb, včetně následujících:

- Aktuální velikost fondů
- Časové objekty, které se v nich strávily
- Čištění oblastí
- Aktualizace připojení

Měli byste také zvážit, jak se ve správci front objeví jediná opěvová relace. Existují vlastnosti továrny připojení k identifikaci aplikace (jako např. `appName`), která by mohla být užitečná.

“použitíIBM MQ classes for JMS” na stránce 69
IBM MQ classes for Java Message Service (IBM MQ classes for JMS) je poskytovatel JMS , který je dodáván s IBM MQ. Stejně jako implementace rozhraní definovaných v balíku javax.jms poskytuje produkt IBM MQ classes for JMS dvě sady rozšíření rozhraní API produktu JMS .

Sdílení připojení TCP/IP v produktu IBM MQ classes for JMS
Je možné vytvořit více instancí kanálu MQI, aby bylo možné sdílet jedno připojení TCP/IP.

Aplikace, které jsou spuštěny ve stejném běhovém prostředí produktu Java a které používají adaptér prostředků produktu IBM MQ classes for JMS nebo IBM MQ k připojení ke správci front pomocí přenosu CLIENT, lze provést tak, aby sdílely stejnou instanci kanálu.

Mezi instancemi kanálu a připojeními TCP/IP existuje vztah 1:1. Jedno připojení TCP/IP je vytvořeno pro každou instanci kanálu.

Je-li kanál definován s parametrem **SHARECNV** nastaveným na hodnotu větší než 1, pak tento počet konverzací může sdílet instanci kanálu. Chcete-li povolit továrnu připojení nebo specifikaci aktivace pro použití této funkce, nastavte vlastnost **SHARECONVALLOWED** na hodnotu YES.

Každé připojení JMS a relace JMS vytvořené aplikací produktu JMS vytváří vlastní konverzaci se správcem front.

Po spuštění specifikace aktivace spustí adaptér prostředků produktu IBM MQ konverzaci se správcem front, který má být použit pro specifikaci aktivace. Každá serverová relace ve fondu relací serveru, která je přidružena ke specifikaci aktivace, také zahájí konverzaci se správcem front.

Atribut SHARECNV je nejlepším přístupem k sdílení připojení. Proto je-li v produktu IBM MQ classes for JMSpoužita hodnota SHARECNV větší než 0, není zaručeno, že nový požadavek na připojení bude vždy sdílet již vytvořené spojení.

Výpočet počtu instancí kanálu

Pomocí následujících vzorců určete maximální počet instancí kanálu vytvořených aplikací:

Specifikace aktivace

Počet instancí kanálu = (*maxPoolDepth_value* + 1)/ *SHARECNV_value*

Kde *maxPoolDepth_value* je hodnota vlastnosti **maxPoolDepth** a *SHARECNV_value* je hodnota vlastnosti **SHARECNV** na kanálu, který je použit specifikací aktivace.

Další aplikace produktu JMS

Počet instancí kanálu = (*jms_connections* + *jms_sessions*)/ *SHARECNV_value*

Kde *jms_connections* je počet připojení vytvořených aplikací, *jms_sessions* je počet relací JMS vytvořených aplikací a *SHARECNV_value* je hodnota vlastnosti **SHARECNV** na kanálu, který je použit specifikací aktivace.

Příklady

Následující příklady ukazují způsob použití vzorců k výpočtu počtu instancí kanálu, které jsou vytvořeny ve správci front aplikacemi s použitím adaptéru prostředků IBM MQ classes for JMS nebo IBM MQ .

Příklad aplikace JMS
Připojení aplikace JMS se připojuje ke správci front pomocí přenosu CLIENT a vytváří připojení JMS a tři relace JMS . Kanál, který aplikace používá pro připojení ke správci front, má vlastnost **SHARECNV** nastavenou na hodnotu 10. Je-li aplikace spuštěna, existují čtyři konverzace mezi aplikací a správcem front a jednou instancí kanálu. Všechny čtyři konverzace sdílejí instanci kanálu.

Příklad specifikace aktivace
Specifikace aktivace se připojuje ke správci front pomocí přenosu CLIENT. Specifikace aktivace je konfigurována s vlastností **maxPoolDepth** nastavenou na hodnotu 10. Kanál, který je specifikace aktivace konfigurován pro použití, má vlastnost **SHARECNV** nastavenou na hodnotu 10. Je-li specifikace aktivace spuštěna a souběžně zpracovává 10 zpráv, počet konverzací mezi specifikací aktivace a správcem front je 11 (10 konverzací pro relace serveru a jedna pro specifikaci aktivace). Počet instancí kanálu, které jsou použity specifikací aktivace, je 2.

Příklad specifikace aktivace

Specifikace aktivace se připojuje ke správci front pomocí přenosu CLIENT. Specifikace aktivace je konfigurována s vlastností **maxPoolDepth** nastavenou na hodnotu 5. Kanál, který je specifikace aktivace konfigurován pro použití, má vlastnost **SHARECNV** nastavenou na hodnotu 0. Je-li specifikace aktivace spuštěna a zpracovává se 5 zpráv souběžně, počet konverzací mezi specifikací aktivace a správcem front je 6 (pět konverzací pro relace serveru a jedna pro specifikaci aktivace). Počet instancí kanálu, které jsou použity aktivační specifikací, je 6, protože vlastnost **SHARECNV** na kanálu je nastavena na 0, každá konverzace používá vlastní instanci kanálu.

Související úlohy

“Určení počtu připojení TCP/IP vytvořených z produktu WebSphere Application Server do produktu IBM MQ” na stránce 462

Produkt IBM WebSphere MQ 7.0 představil novou funkci nazvanou "sdílení konverzací". Pomocí této funkce může více konverzací sdílet instance kanálu MQI, což je také známé jako připojení TCP/IP.

Určení rozsahu portů pro připojení klienta v produktu IBM MQ classes for JMS

Použijte vlastnost LOCALADDRESS k uvedení rozsahu portů, ke kterým se vaše aplikace může vázat.

Pokusí-li se aplikace IBM MQ classes for JMS o připojení ke správci front produktu IBM MQ v režimu klienta, může brána firewall povolit pouze ta připojení, která pocházejí ze zadaných portů nebo z rozsahu portů. V této situaci můžete použít vlastnost LOCALADDRESS objektu ConnectionFactory, továrny QueueConnectionFactory nebo TopicConnectionk určení portu nebo rozsahu portů, na které se aplikace může vázat.

Vlastnost LOCALADDRESS můžete nastavit pomocí nástroje pro administraci produktu IBM MQ JMS , nebo voláním metody setLocalAddress () v aplikaci produktu JMS . Dále je uveden příklad nastavení vlastnosti v rámci aplikace:

```
mqConnectionFactory.setLocalAddress("192.0.2.0(2000,3000)");
```

Když se aplikace připojí ke správci front následně, aplikace se připojí k lokální adrese IP a číslu portu v rozsahu 192.0.2.0(2000) na 192.0.2.0(3000).

V systému s více než jedním síťovým rozhraním můžete také použít vlastnost LOCALADDRESS k uvedení, které síťové rozhraní musí být použito pro připojení.

V případě připojení v reálném čase ke zprostředkovateli je vlastnost LOCALADDRESS relevantní pouze při použití výběrového vysílání. V takovém případě můžete použít vlastnost k určení, které lokální síťové rozhraní musí být použito pro připojení, ale hodnota vlastnosti nesmí obsahovat číslo portu nebo rozsah čísel portů.

Pokud omezíte rozsah portů, může dojít k chybám připojení. Dojde-li k chybě, je vyvolána výjimka JMSException s vloženou výjimku MQException, která obsahuje kód příčiny IBM MQ MQRC_Q_MGR_NOT_AVAILABLE a následující zpráva:

```
Pokus o připojení soketu byl odmítnut kvůli omezením LOCAL_ADDRESS_PROPERTY
```

Může se vyskytnout chyba, pokud jsou použity všechny porty v uvedeném rozsahu, nebo pokud zadaná adresa IP, název hostitele nebo číslo portu nejsou platné (například záporné číslo portu).

Vzhledem k tomu, že produkt IBM MQ classes for JMS může vytvářet jiná připojení než ta, která jsou vyžadována aplikací, vždy zvažte možnost zadání rozsahu portů. Obecně platí, že každá relace vytvořená aplikací vyžaduje jeden port a IBM MQ classes for JMS může vyžadovat tři nebo čtyři další porty. Dojde-li k chybě připojení, zvyšte rozsah portů.

Sdružování připojení, které se v produktu IBM MQ classes for JMSpoužívá ve výchozím nastavení, může mít vliv na rychlost, kterou lze znovu použít porty. V důsledku toho může dojít k chybě připojení, zatímco jsou porty uvolněny.

Komprese kanálu v produktu IBM MQ classes for JMS

Aplikace IBM MQ classes for JMS může použít zařízení produktu IBM MQ ke kompresi záhlaví nebo dat zprávy.

Komprimace dat, která procházejí kanálem IBM MQ , může zlepšit výkon kanálu a snížit provoz sítě. Pomocí funkce dodávané s produktem IBM MQmůžete komprimovat data, která proudí na kanály zpráv a kanály MQI. U obou typů kanálů můžete komprimovat data záhlaví a data zprávy nezávisle na sobě. Standardně nejsou žádná data komprimována na kanálu.

Aplikace produktu IBM MQ classes for JMS určuje techniky, které lze použít pro kompresi dat záhlaví nebo zprávy na připojení vytvořením objektu java.util.Collection . Každá kompresní technika je objekt Integer v kolekci a pořadí, ve kterém aplikace přidá techniky komprese do kolekce, je pořadí, ve kterém jsou techniky komprese vyjednané se správcem front, když aplikace vytvoří připojení. Aplikace pak může předat kolekci do objektu ConnectionFactory voláním metody setHdrCompList(), pro data záhlaví nebo metodou setMsgCompList() pro data zprávy. Je-li aplikace připravena, může vytvořit připojení.

Následující fragmenty kódu popisují popsaný přístup. První fragment kódu ukazuje, jak implementovat kompresi dat záhlaví:

```
Collection headerComp = new Vector();
headerComp.add(new Integer(WMQConstants.WMQ_COMPHDR_SYSTEM));
.
.
.
((MQConnectionFactory) cf).setHdrCompList(headerComp);
.
.
.
connection = cf.createConnection();
```

Druhý fragment kódu ukazuje, jak implementovat kompresi dat zprávy:

```
Collection msgComp = new Vector();
msgComp.add(new Integer(WMQConstants.WMQ_COMPMSG_RLE));
msgComp.add(new Integer(WMQConstants.WMQ_COMPMSG_ZLIBHIGH));
.
.
.
((MQConnectionFactory) cf).setMsgCompList(msgComp);
.
.
.
connection = cf.createConnection();
```

Ve druhém příkladu jsou techniky komprese vyjednané v pořadí RLE, pak ZLIBHIGH, když se vytvoří připojení. Zvolená technika komprese nemůže být změněna během doby životnosti objektu připojení. Chcete-li pro připojení použít kompresi, musí být před vytvořením objektu Connection volány metody setHdrCompList() a setMsgCompList().

Asynchronní vkládání zpráv do produktu IBM MQ classes for JMS
Obvykle, když aplikace odesílá zprávy do místa určení, musí aplikace čekat, až správce front potvrdí, že zpracoval požadavek. Můžete zlepšit výkon systému zpráv za určitých okolností tím, že zvolíte, že nebudete asynchronně vkládat zprávy. Když aplikace asynchronně odešle zprávu, správce front nevrátí úspěch nebo selhání každého volání, ale můžete namísto toho zkontrolovat chyby pravidelně.

Určuje, zda místo určení vrátí řízení aplikaci, aniž by určujete, zda správce front zprávu přijal bezpečně, závisí na následujících vlastnostech:

- Vlastnost cíle JMS PUTASYNCALLOWED (krátký název-PAALD).

  Parametr PUTASYNCALLOWED řídí, zda aplikace produktu JMS mohou asynchronně vkládat zprávy, je-li v základní frontě nebo v tématu reprezentovaném cílem JMS tato volba povolena.
- Vlastnost fronty IBM MQ nebo tématu DEFPRESP (Výchozí typ odezvy put).

  DEFPRESP určuje, zda aplikace, které vložila zprávy do fronty nebo publikují zprávy do daného tématu, mohou využívat funkčnost asynchronního vložení.

Následující tabulka uvádí možné hodnoty vlastností PUTASYNCALLOWED a DEFPRESP a hodnoty, které jsou vyžadovány pro funkčnost asynchronního vkládání, jsou-li povoleny:

Tabulka 46. Vlastnosti PUTASYNCALLOWED a DEFPRESP určují, zda jsou zprávy vsazeny asynchronně.			
JMS Vlastnost cíle	**PUTASYNCALLOWED = NE**	**PUTASYNCALLOWED = ANO**	**PUTASYNCALLOWED = AS_DEST nebo AS_Q_DEF nebo AS_T_DEF**
Vlastnost fronty produktu IBM MQ			
DEFPRESP=SYNCHRONIZACE	Funkčnost asynchronního vložení není povolena	Povolena funkčnost asynchronního vkládání	Funkčnost asynchronního vložení není povolena
DEFPRESP=ASYNC	Funkčnost asynchronního vložení není povolena	Povolena funkčnost asynchronního vkládání	Povolena funkčnost asynchronního vkládání

Pro zprávy odeslané v relaci transakce nakonec aplikace určí, zda správce front přijal zprávy bezpečně při volání produktu `commit()`.

Pokud aplikace odesílá trvalé zprávy v rámci relace s transakcemi a jedna nebo více zpráv není přijato bezpečně, transakce se nezdaří a vygeneruje výjimku. Pokud však aplikace odešle přechodné zprávy v rámci relace s transakcemi a jedna nebo více zpráv není bezpečně doručeno, transakce se úspěšně potvrdí. Aplikace neobdrží žádnou zpětnou vazbu, že přechodné zprávy nedorazly bezpečně.

Pro přechodné zprávy odeslané v relaci, která není součástí transakce, určuje vlastnost SENDCHECKCOUNT objektu *ConnectionFactory* , kolik zpráv má být odesláno, než produkt IBM MQ classes for JMS kontroluje, zda správce front obdržel zprávy bezpečně.

Pokud kontrola zjistí, že jedna nebo více zpráv nebyla bezpečně přijata a aplikace zaregistrovala modul listener výjimek s připojením, produkt IBM MQ classes for JMS volá metodu `onException()` modulu listener výjimek, aby předal aplikaci výjimku JMS .

Výjimka JMS má kód chyby JMSWMQ0028 a tento kód zobrazuje tuto zprávu:

```
At least one asynchronous put message failed or gave a warning.
```

Výjimka JMS má také připojenou výjimku, která poskytuje více podrobností. Výchozí hodnota vlastnosti SENDCHECKCOUNT je nula, což znamená, že žádné takové kontroly nejsou provedeny.

Tato optimalizace má největší přínos pro aplikaci, která se připojuje ke správci front v režimu klienta, a potřebuje odeslat posloupnost zpráv v rychlém sledu, ale nevyžaduje okamžitou zpětnou vazbu od správce front pro každou odeslanou zprávu. Aplikace však přesto může tuto optimalizaci použít i v případě, že se připojuje ke správci front v režimu vazeb, ale očekávaný přínos výkonu není tak velký.

Použití dopředného čtení s IBM MQ classes for JMS
Funkčnost čtení napřed poskytovaná produktem IBM MQ umožňuje netrvalé zprávy, které jsou přijaté mimo transakci, odeslány do produktu IBM MQ classes for JMS před tím, než je aplikace vyžádá. Server IBM MQ classes for JMS ukládá zprávy ve vnitřní vyrovnávací paměti a předává zprávy aplikaci, když se o ně aplikace požádá.

Aplikace produktu IBM MQ classes for JMS , které používají produkt `MessageConsumers` nebo `MessageListeners` k příjmu zpráv z místa určení mimo transakci, mohou využívat funkce čtení napřed. Použití dopředného čtení umožňuje aplikacím, které tyto objekty využívají, aby mohly těžit ze zlepšení výkonu při příjmu zpráv.

To, zda aplikace, která používá produkt `MessageConsumers` nebo `MessageListeners` , může dopředné čtení použít, závisí na následujících vlastnostech:

- JMS Cílová vlastnost READAHEADALLOWED (krátký název-RAALD). READAHEADALLOWED řídí, zda aplikace produktu JMS mohou při získávání nebo procházení přechodných zpráv mimo transakci používat dopředné čtení při získávání nebo procházení netrvalých zpráv mimo transakci, je-li tato volba povolena, je-li základní fronta nebo téma, které je určeno cílem JMS , povoleno.

- Vlastnost fronty IBM MQ nebo tématu DEFREADA (Výchozí dopředné čtení). DEFREADA určuje, zda aplikace, které přijímají nebo prohledávaní dočasné zprávy mimo transakci, mohou použít dopředné čtení.

Následující tabulka uvádí možné hodnoty vlastností READAHEADALLOWED a DEFREADA a jaké hodnoty jsou vyžadovány pro funkce čtení napřed, které mají být povoleny:

Tabulka 47. Vlastnosti READAHEADALLOWED a DEFREADA určuje, zda je při příjmu nebo procházení dočasných zpráv mimo transakci použito dopředné čtení.

Vlastnost místa určení IBM MQ	**READAHEADALLOWED = ANO**	**READAHEADALLOWED = NO**	**AS_DEST nebo AS_Q_DEF nebo AS T_DEF**
Vlastnost fronty produktu IBM MQ			
DEFREADA = NE	Povolena funkčnost čtení napřed	Funkce dopředného čtení není povolena	Funkce dopředného čtení není povolena
DEFREADA = ANO	Povolena funkčnost čtení napřed	Funkce dopředného čtení není povolena	Povolena funkčnost čtení napřed
DEFREADA = VYPNUTO	Funkce dopředného čtení není povolena	Funkce dopředného čtení není povolena	Funkce dopředného čtení není povolena

Je-li povolena funkce dopředného čtení, je-li `MessageConsumer` nebo `MessageListener` vytvořena aplikací, IBM MQ classes for JMS vytvoří vnitřní vyrovnávací paměť pro cíl, který je monitorován produktem `MessageConsumer` nebo `MessageListener` . Pro každý `MessageConsumer` nebo `MessageListener`existuje jedna vnitřní vyrovnávací paměť. Správce front začne odesílat netrvalé zprávy do serveru IBM MQ classes for JMS , když aplikace volá jednu z následujících metod:

- `MessageConsumer.receive()`
- `MessageConsumer.receive(long timeout)`
- `MessageConsumer.receiveNoWait()`
- `Session.setMessageListener(MessageListener listener)`

Produkt IBM MQ classes for JMS automaticky vrátí první zprávu zpět do aplikace, metodou volání metody, kterou provedla aplikace. Ostatní netrvalé zprávy jsou uloženy produktem IBM MQ classes for JMS ve vnitřní vyrovnávací paměti, která byla vytvořena pro místo určení. Když aplikace požádá o další zprávu ke zpracování, vrátí IBM MQ classes for JMS další zprávu ve vnitřní vyrovnávací paměti.

IBM MQ classes for JMS požaduje od správce front další přechodné zprávy, je-li vnitřní vyrovnávací paměť prázdná.

Vnitřní vyrovnávací paměť, kterou používá IBM MQ classes for JMS , se odstraní, když aplikace uzavře `MessageConsumer`nebo JMS relace, ke které je přidružená `MessageListener` .

Pro systém `MessageConsumers`se všechny nezpracované zprávy v interní vyrovnávací paměti ztratí.

Při použití produktu `MessageListeners`, to, co se stane s zprávami v interní vyrovnávací paměti, závisí na vlastnosti cíle JMS READAHEADCLOSEPOLICY (krátký název-RACP). Výchozí hodnota vlastnosti je DELIVER_ALL, což znamená, že relace JMS , která byla použita k vytvoření `MessageListener` , není zavřena, dokud se do aplikace doručí všechny zprávy v interní vyrovnávací paměti. Je-li vlastnost nastavena na DELIVER_CURRENT, bude po zpracování aktuální zprávy aplikací ukončena relace JMS a všechny zbývající zprávy v interní vyrovnávací paměti budou zrušeny.

Zachovaná publikování v produktu IBM MQ classes for JMS
Klient produktu IBM MQ classes for JMS může být konfigurován tak, aby používal zachované publikace.

Vydavatel může určit, že kopie publikace musí být uchována tak, aby mohla být odeslána na budoucí odběratele, kteří se zajímají o dané téma. Tento proces se provádí v produktu IBM MQ classes for JMS nastavením celočíselné vlastnosti JMS_IBM_RETAIN na hodnotu 1. Pro tyto hodnoty byly

definovány konstanty v rozhraní com.ibm.msg.client.jms.JmsConstants . Pokud jste například vytvořili zprávu *msg*a nastavili ji jako zachované publikování, použijte následující kód:

```
// set as a retained publication
msg.setIntProperty(JmsConstants.JMS_IBM_RETAIN, JmsConstants.RETAIN_PUBLICATION);
```

Nyní můžete odeslat zprávu jako normální. V přijaté zprávě lze obdržet dotaz na službu JMS_IBM_RETAIN. Je tedy možné se dotázat, zda je přijatá zpráva zachovaná publikace.

Podpora XA v produktu IBM MQ classes for JMS

Produkt JMS podporuje transakce kompatibilní se standardem XA ve vazbách a v režimu klienta s podporovaným správcem transakcí v rámci kontejneru JEE .

Pokud v prostředí aplikačního serveru vyžadujete funkčnost XA, je třeba aplikaci odpovídajícím způsobem nakonfigurovat. Informace o tom, jak konfigurovat aplikace k použití distribuovaných transakcí, naleznete v dokumentaci k aplikačnímu serveru.

Správce front produktu IBM MQ nemůže vystupovat jako správce transakcí pro produkt JMS.

Použití funkčnosti produktu JMS 2.0

Produkt JMS 2.0 zavádí několik nových oblastí funkčnosti produktu IBM MQ classes for JMS.

Pokud vyvíjíte aplikaci produktu JMS pro produkt IBM MQ 8.0 nebo novější, může být zapotřebí zvážit dopad této funkčnosti na správce front.

Související informace

Rozhraní jazyka produktu IBM MQ Java

JMS 2.0 prodleva doručení

Pomocí produktu JMS 2.0můžete určit prodlevu doručení při odesílání zprávy. Správce front tuto zprávu nedoručí, dokud neuplyne zadaná prodleva doručení.

Aplikace může určit prodlevu doručení v milisekundách, po odeslání zprávy pomocí produktu `MessageProducer.setDeliveryDelay(long deliveryDelay)` nebo `JMSProducer.setDeliveryDelay(long deliveryDelay)`. Tato hodnota se přidá k času, kdy je zpráva odeslána, a dává nejdřívější čas, kdy může každá jiná aplikace tuto zprávu získat.

V produktu IBM MQ 8.0 a novějším je zpoždění doručení implementováno pomocí jedné interní pracovní fronty. Zprávy, které mají nenulovou prodlevu doručení, jsou umístěny do této fronty se záhlavím, které označuje prodlevu doručení a informace o cílové frontě. Komponenta správce front, která se nazývá procesor s prodlevou doručení, monitoruje zprávy ve stavové frontě. Po dokončení prodlevy doručení zprávy je zpráva odsunována z fronty fázování a umístěna do cílové fronty.

Klienti systému zpráv

Implementace IBM MQ prodlevy doručení je k dispozici pouze v případě, že používáte klienta JMS . Používáte-li zpoždění doručení s produktem IBM MQ, platí následující omezení. Tato omezení platí také pro `MessageProducers` a `JMSProducers`, ale `JMSRuntimeExceptions` je hozena v případě `JMSProducers`.

- Jakýkoliv pokus o volání `MessageProducer.setDeliveryDelay` s nenulovou hodnotou při připojení ke správci front dříve než IBM MQ 8.0vede k výsledku `JMSException` se zprávou MQRC_FUNCTION_NOT_SUPPORTED.
- Zpoždění doručení není podporováno pro klastrovaná místa určení s hodnotou **DEFBIND** jinou než MQBND_BIND_NOT_FIXED. Pokud má `MessageProducer` nenulovou prodlevu doručení a je proveden pokus o odeslání do místa určení, které tento požadavek nesplňuje, pak volání skončí v `JMSException` se zprávou MQRC_OPTIONS_ERROR.
- Jakýkoli pokus o nastavení doby životnosti, která je menší než dříve zadaná nenulová prodleva doručení nebo naopak, má za následek `JMSException` s chybovou zprávou MQRC_EXPIRY_ERROR.

Tato kontrola se provádí při volání metod `setTimeToLive` nebo `setDeliveryDelay` nebo `send` , v závislosti na konkrétní sadě vybraných operací.

- Použití zachovaných publikování a zpoždění doručení není podporováno. Probíhá pokus o publikování zprávy s prodlevou doručení, pokud byla tato zpráva označena jako zachovaná s použitím `msg.setIntProperty(JmsConstants.JMS_IBM_RETAIN, JmsConstants.RETAIN_PUBLICATION)` výsledků v produktu `JMSException` se zprávou MQRC_OPTIONS_ERROR.
- Zpoždění doručení a seskupování zpráv není podporováno a každý pokus o použití této kombinace vede k výjimce JMSException s chybovou zprávou MQRC_OPTIONS_ERROR.

Jakékoli selhání při odeslání zprávy s prodlevou doručení způsobí, že klient zahodí `JMSException` s vhodnou chybovou zprávou, například plnou frontu. V některých situacích se může chybová zpráva vztahovat na cílové místo určení nebo na výstupní frontu nebo obojí.

Poznámka: Produkt IBM MQ umožňuje aplikacím, které vložila zprávu do pracovní jednotky, znovu získat stejnou zprávu i přesto, že se nepotvrdila jednotka práce. Tato technika nefunguje s prodlevou doručení, protože zpráva není umístěna do pracovní fronty, dokud není potvrzena jednotka práce, a proto nebyla odeslána do cílového místa určení.

Autorizace

Produkt IBM MQ provádí kontroly autorizace v původním cílovém místě určení, když aplikace odesílá zprávu s nenulovým zpožděním doručení. Není-li aplikace autorizována, dojde k selhání odeslání. Když správce front zjistí, že je prodleva doručení zprávy dokončena, otevře cílovou frontu. V tomto okamžiku nejsou prováděny žádné kontroly autorizace.

SYSTEM.DDELAY.LOCAL.QUEUE

Fronta systému, SYSTEM.DDELAY.LOCAL.QUEUEse používá k implementaci prodlevy doručení.

- Multi V systémech Multiplatformy, SYSTEM.DDELAY.LOCAL.QUEUE ve výchozím nastavení existuje. Systémová fronta musí být změněna tak, aby její atributy MAXMSGL a MAXDEPTH byly dostatečné pro očekávané zatížení.
- z/OS V systémech IBM MQ for z/OS, SYSTEM.DDELAY.LOCAL.QUEUE se používá jako fronta fázování pro zprávy, které jsou odesílány s prodlevou doručení na lokální i sdílené fronty. V systému z/OSmusí být fronta vytvořena a musí být definována tak, aby byly atributy MAXMSGL a MAXDEPTH dostatečné pro očekávané zatížení.

Je-li tato fronta vytvořena, musí být zabezpečena tak, aby k ní bylo možné přistupovat co nejméně málo uživatelů. Přístup do fronty musí být určen pouze pro účely údržby a monitorování.

Je-li zpráva odeslána aplikací JMS s nenulovým zpožděním doručení, je vložena do této fronty s novým ID zprávy. ID původní zprávy se umístí do ID korelace zprávy. Toto ID korelace umožňuje aplikaci v případě potřeby načíst zprávu z pracovní fronty, například pokud byla omylem použita velká prodleva doručení.

Aspekty pro produkt z/OS

z/OS

Je-li váš systém spuštěn v systému z/OS, je třeba vzít v úvahu další aspekty, pokud chcete použít prodlevu doručení.

Má-li být použito zpoždění doručení, fronta systému SYSTEM.DDELAY.LOCAL.QUEUE musí být definována. Musí být definován s třídou paměti, která je dostatečná pro jeho očekávané zatížení, a s uvedeným identifikátorem INDXTYPE (NONE) a MSGDLVSQ (FIFO). Ukázková definice systémové fronty je poskytnuta, komentovaný, v JCL CSQ4INSG .

Prodleva doručení není chráněna OPMODE. Pokud použijete prodlevu doručení se správcem front produktu IBM MQ 8.0 a pak provedete migraci zpět na předchozí vydání, všechny zprávy v systému SYSTEM.DDELAY.LOCAL.QUEUE jsou zakonány, pokud s nimi ručně nevypořádujete ručně.

Sdílené fronty

Zpoždění doručení je podporováno pro odesílání zpráv do sdílených front. Existuje však pouze jedna soukromá fázová fronta, která se použije bez ohledu na to, zda je cílová fronta sdílená či nikoli. Správce front, který vlastní tuto soukromou frontu, musí být spuštěn tak, aby při dokončení prodlevy odeslala zpožděnou zprávu do cílové sdílené fronty.

Poznámka: Pokud je netrvalá zpráva vložena s odložením doručení do sdílené fronty a správce front, který vlastní pracovní frontu, se ukončí, původní zpráva se ztratí. Vzhledem k tomu, že přechodné zprávy odeslané s prodlevou doručení do sdílené fronty budou pravděpodobně ztraceny než netrvalé zprávy odeslané bez prodlevy doručení do sdílené fronty.

Rozlišení cílového místa určení

Je-li zpráva odeslána do fronty, je rozlišení dvakrát řízeno aplikací JMS a jednou správcem front, když převezme zprávu z pracovní fronty a odešle ji do cílové fronty.

Cílové odběry pro publikování se shodují, když aplikace JMS volá metodu odeslání.

Je-li zpráva odeslána s perzistencí nebo s prioritou podle definice fronty, pak je hodnota nastavena na prvním rozlišení a ne na druhou.

Interval vypršení

Zpoždění doručení zachovává chování vlastnosti vypršení platnosti, MQMD.Expiry. Pokud byla například zpráva vložena z aplikace JMS s intervalem vypršení 20 000 ms a s prodlevou doručení 5000 ms a po uplynutí uplynulé doby 10.000 ms, může být hodnota pole MQMD.expiry přibližně 50 desetin sekundy. Tato hodnota označuje, že uplynulo 15 sekund od doby, kdy byla zpráva vložena, do doby, kdy byla zpráva odeslána.

Pokud dojde k vypršení platnosti zprávy během pracovní fronty a jedné z voleb MQRO_EXPIRATION_ *, bude vygenerovaná sestava použita pro původní zprávu odesílanou aplikací, záhlaví použité pro uchování informací o zpoždění doručení je odebráno.

Zastavení a spuštění procesoru s prodlevou doručení

z/OS V systému z/OSje procesor s prodlevou doručení integrován do adresního prostoru MSTR správce front. Když se správce front spustí, spustí se také procesor s prodlevou doručení. Je-li pracovní fronta k dispozici, otevře frontu a čeká na doručení zpráv, které mají být zpracovány. Pokud nebyla pracovní fronta definována nebo je zakázána pro získání, nebo dojde k jiné chybě, procesor s prodlevou doručení se ukončí. Pokud je výstupní fronta později definována nebo změněna na povolení, dojde k restartu procesoru s prodlevou při doručení. Pokud se procesor s prodlevou doručení vypne z jakékoli jiné příčiny, lze jej restartovat změnou atributu PUT pracovní fronty z hodnoty ENABLED na hodnotu DISABLED a znovu zpět na ENABLED . Potřebujete-li z jakéhokoli důvodu zastavit procesor s prodlevou doručení, nastavte atribut PUT pracovní fronty na DISABLED.

Multi V systému Multiplatformyse procesor s prodlevou spustí se správcem front a je automaticky restartován v případě zotavitelné poruchy.

Selhání při vložení do cílové fronty

Pokud nelze pozdržet zprávu do cílové fronty po jejím dokončení, je zpráva řešena tak, jak je uvedeno ve svých volbách sestavy: buď je vyřazena, nebo odeslána do fronty nedoručených zpráv. Dojde-li k selhání této akce, bude proveden pokus o vložení zprávy později. Je-li akce úspěšná, vygeneruje se zpráva výjimky a odešle se do uvedené fronty, je-li požadována sestava. Pokud nebyla zpráva sestavy odeslána, odešle se zpráva do fronty nedoručených zpráv. Pokud se odeslání sestavy do fronty nedoručených zpráv nezdaří a zpráva je trvalá, všechny změny budou vyřazeny a původní zpráva odvolána a znovu doručena později. Pokud se zpráva nestálá, zpráva sestavy je vyřazena, ale jsou potvrzeny jiné změny. Pokud odložené publikování nelze doručit, protože odběratel má zrušení odběru nebo v případě trvalého

odběratele, protože se odpojil, pak je zpráva vyřazena bezobslužně. Zprávy sestav se stále generují, jak je popsáno dříve.

Pokud odložené publikování nelze doručit odběrateli a místo toho se umístí do fronty nedoručených zpráv, a do fronty nedoručených zpráv dojde k selhání, je zpráva vyřazena.

Pro snížení pravděpodobnosti vložení do cílové fronty po dokončení prodlevy doručení provádí správce front některé základní kontroly, když klient JMS odešle zprávu s nenulovým zpožděním doručení. Tyto kontroly zahrnují, zda je fronta zakázaná, je-li zpráva větší než maximální povolená délka zprávy, a pokud je fronta plná.

Publikování/odběr

Když aplikace JMS odešle zprávu s nenulovým doručením o doručení, dojde k porovnání publikování s dostupnými odběry. Zpráva pro každého odpovídajícího odběratele je vložena do systému SYSTEM.DDELAY.LOCAL.QUEUE frontu, kde je uchována, dokud se zpoždění doručení nedokončí. Je-li některý z těchto odběratelů proxy odběru pro jiného správce front, bude po dokončení zpoždění doručení provedeno odhlášení od tohoto správce front. To může vést k tomu, že odběratelé na jiném správci front obdrží publikace, které byly původně publikovány dříve, než se přihlásili k odběru. Jedná se o odchylku od specifikace JMS 2.0 .

Zpoždění doručení s publikováním/odběrem je podporováno pouze v případě, že je cílové téma nakonfigurováno pomocí (N) PMSGDLV = ALLAVAIL. Pokus o použití jakýchkoli jiných hodnot má za následek chybu MQRC_PUBLICATION_FAILURE. Pokud při vložení zprávy do cílové fronty selže procesor s prodlevou doručení, výsledek je popsán v sekci "Selhání při vložení do cílové fronty".

Hlášení zpráv

Všechny volby sestavy jsou podporovány a jsou podporovány doručovacím procesorem jiné než následující volby, které jsou ignorovány, ale jsou předány ve zprávě, když je odeslána do cílové fronty:

- MQRO_COA *
- MQRO_COD *
- MQRO_PAN/MQRO_NAN
- AKTIVITA MQRO_ACTIVITY

Klonované a sdílené odběry
V produktu IBM MQ 8.0 nebo novějším existují dvě metody, jak poskytnout více spotřebitelům přístup ke stejnému odběru. Tyto dvě metody jsou buď pomocí klonovaných odběrů, nebo pomocí sdílených odběrů.

Klonování odběrů

Klonovaný odběr je rozšíření produktu IBM MQ . Klonované odběry umožňují více konzumentům v různých prostředích JVM (Java Virtual Machine) souběžně s přístupem k odběru. Toto chování lze použít k nastavení vlastnosti **CLONESUPP** na hodnotu `Enabled` na objektu connectionFactory . Ve výchozím nastavení je **CLONESUPP** `Zakázáno`. Klonované odběry lze povolit pouze u trvalých odběrů. Je-li volba **CLONESUPP** povolena, každé následné připojení vytvořené pomocí této connectionFactory je naklonováno.

Trvalý odběr lze považovat za klonován, pokud je vytvořen jeden nebo více spotřebitelů pro příjem zpráv z tohoto odběru, tj. byly vytvořeny uvedením stejného názvu odběru. To lze provést pouze v případě, že připojení, pod kterým byly vytvořeny spotřebitelé, má **CLONESUPP** nastaveno na `Povoleno` na MQConnectionFactory. Je-li publikována zpráva v tématu odběru, odešle se kopie této zprávy do odběru. Tato zpráva je k dispozici všem spotřebitelům, ale pouze jedna z nich přijímá.

Poznámka: Povolení klonovaných odběrů rozšiřuje specifikaci JMS .

Sdílené odběry

Specifikace JMS 2.0 zavádí sdílené odběry, které umožňují sdílení zpráv z odběru tématu mezi více spotřebiteli. Každá zpráva z odběru je doručena pouze jednomu ze spotřebitelů v daném odběru. Sdílené odběry jsou povoleny příslušným voláním rozhraní API produktu JMS 2.0 .

Rozhraní API lze volat jedním z následujících způsobů:

- Z aplikace Java SE (nebo kontejneru klienta Java EE).
- Z servletu nebo implementace objektu MDB.

Specifikace JMS 2.0 nedefinuje žádný standardní způsob řízení objektu MDB z sharedSubscription, takže produkt IBM MQ 8.0 nebo pozdější poskytuje vlastnost specifikace aktivace sharedSubscription pro tento účel. Další informace o této vlastnosti viz "Konfigurace adaptéru prostředků pro příchozí komunikaci" na stránce 415 a "Příklady toho, jak definovat vlastnost sharedSubscription" na stránce 430.

Je-li povolen sdílený odběr, nelze jej zrušit sdílení.

Sdílené odběry lze vytvořit buď jako trvalé, nebo bez trvalého odběru. Neexistuje žádný požadavek na samostatné vytvoření objektů na straně správce front, které přesahují normální konfiguraci JMS , všechny vyžadované objekty jsou vytvářeny dynamicky.

Rozhodování mezi sdílenými nebo klonovanými odběry

Když určujete, zda použít sdílené nebo naklonované odběry, zvažte výhody obou. Je-li to možné, používejte sdílené odběry, protože jsou to definované chování specifikace, spíše než rozšíření specifické pro produkt IBM MQ .

Následující tabulka obsahuje některé body, které je třeba vzít v úvahu při rozhodování mezi sdílenými a klonovanými odběry:

Tabulka 48. Pokyny pro výběr mezi sdílenými odběry a klonovanými odběry

Sdílené odběry	**Klonované odběry**
Sdílené odběry jsou standardní součástí specifikace JMS 2.0 .	Klonování odběrů je specifické rozšíření produktu IBM MQ .
Sdílené odběry se vytvářejí pomocí explicitních volání metod rozhraní API.	Klonované odběry jsou řízeny administrativně na úrovni ConnectionFactory .
Sdílené odběry mohou být trvalé nebo netrvalé.	Klonované odběry mohou být pouze trvalé.
Sdílené odběry jsou explicitně vytvářeny na základě jednotlivých odběrů.	Klonované odběry se používají pro všechny trvalé odběry v rámci připojení, pro které je funkce povolena.
Je-li odběr vytvořen jako sdílený, nelze jej později změnit na nesdílený, nebo naopak.	Je možné změnit odběr z klonovaného na neklonité při každém opětovném otevření, pokud se změnila vlastnost **CLONESUPP** vlastnícího připojení.

Pokusí se o změnu ostrosti existujícího odběru

V9.0.0.1

Je-li odběr vytvořen jako sdílený, nelze jej později změnit na nesdílený, nebo naopak.

V produktu IBM MQ 9.0.0 Fix Pack 1byl správce front produktu IBM MQ aktualizován tak, aby v případě, že aplikace produktu JMS 2.0 vytvořila trvalý nesdílený odběr, a další aplikace mimo aplikaci JMS 2.0 se poté pokusí obnovit odběr, je vrácen následující kód příčiny:

```
2432 (MQRC_SUB_ALREADY_EXISTS)
```

Související odkazy

"Příklady toho, jak definovat vlastnost sharedSubscription" na stránce 430

Vlastnost sharedSubscription specifikace aktivace můžete definovat v rámci souboru WebSphere Application Server Liberty `server.xml` . Jinou možností je definovat vlastnost v rámci objektu typu message-driven bean (MDB) pomocí anotací.

Související informace

Odběratelé a odběry

Trvalost odběru

Použití sdílených odběrů JMS 2.0

CLONESUPP

Vlastnost SupportMQExtensions

Specifikace JMS 2.0 zavádí změny v chování některých způsobů chování. Produkt IBM MQ 8.0 a novější zahrnuje vlastnost `com.ibm.mq.jms.SupportMQExtensions`, kterou lze nastavit na hodnotu *TRUE*, aby se tato změněná chování vrátila zpět na předchozí implementace.

Tři oblasti funkčnosti jsou vráceny zpět nastavením parametru `SupportMQExtensions` na hodnotu *True*:

Priorita zprávy
: Pro zprávy může být přiřazena priorita, *0* - *9*. Před JMS 2.0mohou zprávy také použít hodnotu *-1*označující, že je použita výchozí priorita fronty. JMS 2.0 nepovoluje nastavení priority zprávy *-1* . Zapnutí `SupportMQExtensions` umožňuje použít hodnotu *-1* .

ID klienta
: Specifikace JMS 2.0 vyžaduje, aby ID klientů bez hodnoty null byla při navázání spojení kontrolována na jedinečnost. Zapnutí `SupportMQExtensions`znamená, že tento požadavek není ignorován a že ID klienta lze použít znovu.

NoLocal
: Specifikace JMS 2.0 vyžaduje, aby při zapnutí této konstanty nemohl spotřebitel přijímat zprávy, které jsou publikovány se stejným ID klienta. Před JMS 2.0byl tento atribut nastaven na odběrateli, aby zabránil příjmu zpráv, které jsou publikovány vlastním připojením. Zapnutí produktu `SupportMQExtensions` vrátí toto chování na její předchozí implementaci.

Vlastnost `com.ibm.mq.jms.SupportMQExtensions` je logická vlastnost obsažená v souboru `com.ibm.mqjms.jar`. Tuto vlastnost lze nastavit následujícím způsobem:

```
java -Dcom.ibm.mq.jms.SupportMQExtensions=true
```

Tuto vlastnost lze nastavit buď jako standardní systémovou vlastnost prostředí JVM u příkazu **`java`** , nebo může být obsažena v konfiguračním souboru IBM MQ classes for JMS .

Související pojmy

“Konfigurační soubor IBM MQ classes for JMS” na stránce 81

Konfigurační soubor IBM MQ classes for JMS určuje vlastnosti, které se používají ke konfiguraci produktu IBM MQ classes for JMS.

Související odkazy

“Vlastnosti použité ke konfiguraci chování klienta produktu JMS” na stránce 88

Tyto vlastnosti slouží ke konfiguraci chování klienta produktu JMS .

IBM MQ classes for JMS Zařízení aplikačního serveru

Toto téma popisuje, jak produkt IBM MQ classes for JMS implementuje třídu ConnectionConsumer a rozšířenou funkčnost ve třídě relace. Shrnuje také souhrn funkce fondu relací serveru.

Důležité: Tyto informace slouží pouze pro referenci. Aplikace nesmí být napsána pro použití tohoto rozhraní: je používána v rámci adaptéru prostředků produktu IBM MQ pro připojení k serverům Java EE. Praktické informace o připojení najdete v tématu “Použití adaptéru prostředků produktu IBM MQ” na stránce 400.

Produkt IBM MQ classes for JMS podporuje zařízení ASF (Application Server Facilities), které jsou specifikovány ve specifikaci *Java Message Service Specification* (viz Oracle Technology Network for Java Developers). Tato specifikace identifikuje tři role v rámci tohoto programovacího modelu:

- **Poskytovatel JMS** poskytuje funkce ConnectionConsumer a rozšířené funkce relace.
- **Aplikační server** poskytuje funkce ServerSessionPool a ServerSession .
- **Klientská aplikace** používá funkčnost, kterou poskytuje poskytovatel JMS a dodávka aplikačního serveru.

Informace v tomto tématu se nepoužijí, pokud aplikace používá v reálném čase připojení ke zprostředkovateli.

JMS ConnectionConsumer

Rozhraní ConnectionConsumer poskytuje výkonnou metodu k souběžnému doručování zpráv do fondu podprocesů.

Specifikace JMS umožňuje aplikačnímu serveru úzce spolupracovat s implementací produktu JMS prostřednictvím rozhraní produktu `ConnectionConsumer` . Tato funkce poskytuje souběžné zpracování zpráv. Aplikační server obvykle vytvoří fond podprocesů a implementace produktu JMS zpřístupní tyto zprávy těmto podprocesům. Aplikační server se systémem JMS(například WebSphere Application Server) může tuto funkci použít k zajištění funkčnosti systému zpráv na vysoké úrovni, jako jsou například objekty bean řízené zprávami.

Běžné aplikace nepoužívají ConnectionConsumer, ale odborníci JMS ji mohou používat. Pro takové klienty nabízí ConnectionConsumer výkonnou metodu k souběžnému doručování zpráv do fondu podprocesů. Když zpráva dorazí do fronty nebo tématu, produkt JMS vybere vlákno z fondu a doručí do ní dávku zpráv. Chcete-li to provést, produkt JMS spustí přidruženou metodu `onMessage()` metody MessageListener.

Stejného efektu lze dosáhnout vytvořením více objektů Session a MessageConsumer , přičemž každý má registrovaný objekt MessageListener. Hodnota ConnectionConsumer však poskytuje lepší výkon, menší využití prostředků a větší flexibilitu. Požaduje se zejména, že je potřeba méně objektů relace.

Plánování aplikace pomocí ASF

Tento oddíl popisuje, jak naplánovat aplikaci zahrnující:

- “Obecné zásady pro výměnu zpráv mezi dvěma body pomocí ASF” na stránce 295
- “Obecné zásady pro publikování/odběr systému zpráv pomocí ASF” na stránce 296
- “Odebrání zpráv z fronty v ASF” na stránce 297
- Ošetřování škodlivých zpráv v ASF. Viz “Zpracování nezpracovatelných zpráv v produktu IBM MQ classes for JMS” na stránce 203.

Obecné zásady pro výměnu zpráv mezi dvěma body pomocí ASF

Toto téma obsahuje obecné informace o systému zpráv typu point-to-point s použitím ASF.

Když aplikace vytvoří objekt ConnectionConsumer z objektu QueueConnection , určuje objekt fronty JMS a řetězec selektoru. Volba ConnectionConsumer poté začne poskytovat zprávy pro relace v přidružené oblasti ServerSession. Zprávy dorazí do fronty a pokud se shodují s selektorem, jsou doručeny do relací v přidružené oblasti ServerSession.

Za IBM MQ podmínek se objekt fronty odkazuje buď na QLOCAL, nebo na QALIAS na lokálním správci front. Je-li to QALIAS, musí se QALIAS odkazovat na QLOCAL. Plně vyřešený IBM MQ QLOCAL je známý jako *základní QLOCAL*. ConnectionConsumer má být *aktivní* , pokud není zavřen a jeho nadřízený objekt QueueConnection je spuštěn.

Je možné, aby pro více ConnectionConsumers, každá s různými selektory, bylo spuštěno na stejném podkladovém prostředí QLOCAL. Chcete-li zachovat výkonnost, nesmí se nežádoucí zprávy hromadit ve frontě. Nechtěné zprávy jsou ty, pro které nemá žádný aktivní ConnectionConsumer odpovídající selektor. Továrnu QueueConnectionmůžete nastavit tak, aby byly tyto nežádoucí zprávy odebrány z fronty (podrobnosti viz “Odebrání zpráv z fronty v ASF” na stránce 297). Toto chování můžete nastavit jedním ze dvou způsobů:

- Pomocí administračního nástroje produktu JMS nastavte továrnu QueueConnectionna hodnotu MRET (NO).
- Ve vašem programu použijte:

```
MQQueueConnectionFactory.setMessageRetention(WMQConstants.WMQ_MRET_NO)
```

Pokud toto nastavení nezměníte, bude pro výchozí nastavení zachovány tyto nežádoucí zprávy ve frontě.

Při nastavování správce front produktu IBM MQ vezměte v úvahu následující body:

- Základní hodnota QLOCAL musí být povolena pro sdílený vstup. Chcete-li to provést, použijte následující příkaz MQSC:

```
ALTER QLOCAL( your.qlocal.name ) SHARE GET(ENABLED)
```

- Váš správce front musí mít povolenu frontu nedoručených zpráv. Pokud má vlastnost ConnectionConsumer problém, když umístí zprávu do fronty nedoručených zpráv, doručení zprávy ze základní fronty QLOCAL se zastaví. Chcete-li definovat frontu nedoručených zpráv, použijte:

```
ALTER QMGR DEADQ( your.dead.letter.queue.name )
```

- Uživatel, který spouští objekt ConnectionConsumer , musí mít oprávnění k provedení operace MQOPEN s MQOO_SAVE_ALL_CONTEXT a MQOO_PASS_ALL_CONTEXT. Podrobnosti naleznete v dokumentaci produktu IBM MQ pro příslušnou platformu.
- Pokud jsou nežádoucí zprávy ponechány ve frontě, sníží se výkon systému. Proto plánujte své selektory zpráv tak, aby se mezi nimi ConnectionConsumers odebrali všechny zprávy z fronty.

Podrobnosti o příkazech MQSC najdete v tématu Příkazy MQSC.

Obecné zásady pro publikování/odběr systému zpráv pomocí ASF
Volba ConnectionConsumers přijímá zprávy pro určené téma. Objekt ConnectionConsumer může být trvalý nebo dočasný. Musíte určit, kterou frontu nebo fronty má ConnectionConsumer používat.

Když aplikace vytvoří objekt ConnectionConsumer z objektu TopicConnection , určuje objekt Topic a řetězec selektoru. Hodnota ConnectionConsumer poté začne přijímat zprávy, které odpovídají selektoru na daném tématu, včetně všech zachovaných publikování pro odebírané téma.

Volitelně může aplikace vytvořit trvalý ConnectionConsumer , který je přidružen ke specifickému názvu. Tento ConnectionConsumer přijímá zprávy, které byly publikovány na téma od poslední aktivace trvanlivého ConnectionConsumer . Obdrží všechny takové zprávy, které se shodují se selektorem na tématu. Pokud však hodnota ConnectionConsumer používá dopředné čtení, může ztratit přechodné zprávy, které se nacházejí ve vyrovnávací paměti klienta při zavření.

Je-li produkt IBM MQ classes for JMS v režimu migrace poskytovatele systému zpráv produktu IBM MQ , je pro netrvalé odběry ConnectionConsumer použita samostatná fronta.Konfigurovatelná volba CCSUB u továrny TopicConnectionurčuje frontu, která má být použita. Obvykle příkaz CCSUB určuje jednu frontu pro použití všemi prvky ConnectionConsumers , které používají stejné Továrny TopicConnection. Je však možné, aby každý ConnectionConsumer generoval dočasnou frontu uvedením prefixu názvu fronty následovaného hvězdičkou (*).

Je-li produkt IBM MQ classes for JMS v režimu migrace poskytovatele systému zpráv produktu IBM MQ , vlastnost CCDSUB daného tématu určuje frontu, která má být použita pro trvalé odběry. Opět platí, že se jedná o frontu, která již existuje, nebo předponu názvu fronty, za kterou následuje hvězdička (*). Uvedete-li frontu, která již existuje, budou všechny trvalé ConnectionConsumers , které se přihlašují k tématu, používají tuto frontu. Uvedete-li předponu názvu fronty následovanou hvězdičkou (*), vygeneruje se fronta poprvé, kdy se vytvoří trvalý ConnectionConsumer s konkrétním názvem. Tato fronta je znovu použita později, je-li vytvořen trvalý objekt ConnectionConsumer se stejným názvem.

Při nastavování správce front produktu IBM MQ vezměte v úvahu následující body:

- Váš správce front musí mít povolenu frontu nedoručených zpráv. Pokud má vlastnost ConnectionConsumer problém, když umístí zprávu do fronty nedoručených zpráv, doručení zprávy ze základní fronty QLOCAL se zastaví. Chcete-li definovat frontu nedoručených zpráv, použijte:

```
ALTER QMGR DEADQ( your.dead.letter.queue.name )
```

- Uživatel, který spouští objekt ConnectionConsumer , musí mít oprávnění k provedení operace MQOPEN s MQOO_SAVE_ALL_CONTEXT a MQOO_PASS_ALL_CONTEXT. Podrobnosti naleznete v dokumentaci produktu IBM MQ pro příslušnou platformu.
- Výkon pro jednotlivé ConnectionConsumer můžete optimalizovat vytvořením samostatné, vyhrazené, fronty pro tuto frontu. Jedná se o náklady na využití dodatečných prostředků.

Odebrání zpráv z fronty v ASF
Když aplikace používá ConnectionConsumers, produkt JMS může potřebovat odebrat zprávy z fronty v řadě situací.

Tyto situace jsou následující:

Chybně formátovaná zpráva
Je možné, že se objeví zpráva, že JMS nelze analyzovat.

Nezpracovatelná zpráva
Zpráva se může dostat do prahové hodnoty vrácení, ale ConnectionConsumer ji neopětuje ve frontě vrácení.

Není zájem ConnectionConsumer
Pro systém zpráv typu point-to-point, je-li továrna QueueConnectionnastavena tak, aby neuchovávaly nežádoucí zprávy, je doručena zpráva, která je nechtěná některým z ConnectionConsumers.

V takových situacích se ConnectionConsumer pokusí o odebrání zprávy z fronty. Volby odebrání v poli sestavy v rámci zprávy MQMD zprávy nastavují přesné chování. Tyto volby jsou:

MQRO_DEAD_LETTER_Q
Zpráva je znovu zařazena do fronty nedoručených zpráv správce front. Toto nastavení je výchozí.

MQRO_DISCARD_MSG
Zpráva byla zrušena.

Volba ConnectionConsumer také vygeneruje zprávu sestavy a to také závisí na poli sestavy MQMD zprávy. Tato zpráva se odešle na zprávu ReplyToQ zprávy na ReplyToQmgr. Pokud dojde k chybě při odesílání zprávy sestavy, bude místo toho odeslána zpráva do fronty nedoručených zpráv. Volby sestavy výjimek v poli sestavy v podrobnostech sestavy MQMD zprávy sestavy. Tyto volby jsou:

VÝJIMKA MQRO_EXCEPTION
Bude vygenerována zpráva sestavy obsahující MQMD původní zprávy. Neobsahuje žádná těla zprávy.

MQRO_EXCEPTION_WIT_DATA
Bude vygenerována zpráva sestavy obsahující deskriptor MQMD, všechny záhlaví MQ a 100 bajtů dat těla.

MQRO_EXCEPTION_WITH_FULL_DATA
Bude vygenerována zpráva sestavy, která obsahuje všechna data z původní zprávy.

default
Negeneruje se žádná zpráva sestavy.

Když se vygenerují zprávy sestav, jsou uznány následující volby:

- MQRO_NEW_MSG_ID
- MQRO_PASS_MSG_ID
- MQRO_COPY_MSG_ID_TO_CORREL_ID
- ID_KOLEKCE_MQRO_PASS_RELACE_

Pokud nelze nezpracovatelnou zprávu zařadit do fronty, pravděpodobně proto, že je fronta nedoručených zpráv plná nebo je-li autorizace nesprávně zadána, záleží na tom, jak bude zpráva přetrvávajícím způsobem závislá. Je-li zpráva přechodná, bude zpráva vyřazena a nebude vygenerována žádná zpráva sestavy. Je-li zpráva trvalá, doručení zpráv všem odběratelům připojení, kteří naslouchají na tomto místě

určení, se zastaví. Tito spotřebitelé připojení musí být zavřeni a problém vyřešen před tím, než je možné znovu vytvořit a restartovat doručení zprávy.

Je důležité definovat frontu nedoručených zpráv a pravidelně kontrolovat, zda nedošlo k žádným problémům. Zejména zajistěte, aby fronta nedoručených zpráv nedosáhla své maximální hloubky a aby maximální velikost zprávy byla dostatečně velká pro všechny zprávy.

Je-li zpráva zařazena do fronty nedoručených zpráv, předchází mu záhlaví nedoručených zpráv IBM MQ (MQDLH). Podrobnosti o formátu rozhraní MQDLH najdete v tématu Záhlaví MQDLH-Dead-letter . Můžete identifikovat zprávy, které objekt ConnectionConsumer umístil do fronty nedoručených zpráv, nebo zprávy sestav, které generoval ConnectionConsumer , a to pomocí následujících polí:

- PutApplTyp je `MQAT_JAVA (0x1C)`
- PutApplNázev je "`MQ JMS ConnectionConsumer`"

Tato pole jsou ve frontě MQDLH zpráv ve frontě nedoručených zpráv a ve zprávách sestav MQMD. Pole zpětné vazby MQMD a pole Příčina operace MQDLH obsahují kód popisující chybu. Podrobné informace o těchto kódech najdete v tématu “Kódy příčiny a zpětné vazby v ASF” na stránce 299. Ostatní pole jsou popsána v části MQDLH-Dead-letter header.

Zpracování nezpracovatelných zpráv v ASF
V rámci služeb aplikačního serveru je zpracování nezpracovatelných zpráv ošetřeno nepatrně jiným způsobem v produktu IBM MQ classes for JMS.

Informace o zpracování nezpracovatelných zpráv v produktu IBM MQ classes for JMSnaleznete v tématu “Zpracování nezpracovatelných zpráv v produktu IBM MQ classes for JMS” na stránce 203.

Když používáte funkce ASF (Application Server Facilities), ConnectionConsumer, nikoli MessageConsumer, zpracuje nezpracovatelné zprávy. Funkce ConnectionConsumer znovu řadí zprávy podle vlastností QName fronty BackoutThreshold a QName BackoutRequeue.

Když aplikace používá ConnectionConsumers, okolnosti, za kterých je zpráva zálohována, závisí na relaci, kterou poskytuje aplikační server:

- Je-li relace netransakční, s parametrem AUTO_ACKNOWLEDGE nebo DUPS_OK_ACKNOWLEDGE, je zpráva vrácena pouze po chybě systému nebo v případě neočekávaného ukončení činnosti aplikace.
- Pokud relace není součástí transakce s CLIENT_ACKNOWLEDGE, nepotvrzené zprávy mohou být vráceny aplikačním serverem voláním Session.recover().

  Obvykle implementace klienta MessageListener nebo aplikační server volá funkci Message.acknowledge(). Message.acknowledge() bere na vědomí všechny zprávy doručené v relaci tak daleko.
- Je-li relace ve funkci transakce, může aplikační server volat nepotvrzené zprávy voláním funkce Session.rollback().
- Pokud aplikační server dodává aplikaci XASession, jsou zprávy potvrzovány nebo zálohovány v závislosti na distribuované transakci. Aplikační server přebírá odpovědnost za dokončení transakce.

Související pojmy
“Zpracování nezpracovatelných zpráv v produktu IBM MQ classes for JMS” na stránce 203
Je nezpracovatelná zpráva, kterou nelze zpracovat přijímající aplikací. Pokud je nezpracovatelná zpráva doručena aplikaci a odvolána, může ji produkt IBM MQ classes for JMS přesunout do fronty vyřazených zpráv.

Ošetření chyb

Tento oddíl pokrývá různé aspekty ošetření chyb, včetně “Obnova z chybových stavů v ASF” na stránce 298 a “Kódy příčiny a zpětné vazby v ASF” na stránce 299.

Obnova z chybových stavů v ASF
Pokud má vlastnost ConnectionConsumer závažnou chybu, bude doručení zprávy všem uživatelům ConnectionConsumers se zájmem o stejné zastavení QLOCAL. Pokud k tomu dojde, bude upozorněn

kterýkoli modul ExceptionListener , který je registrován u ovlivněného připojení. Existují dva způsoby, jak se může aplikace zotavit z těchto chybových stavů.

Typicky se vyskytne závažná chyba v této povaze, pokud ConnectionConsumer nemůže znovu zařadit zprávu do fronty nedoručených zpráv nebo se setká s chybou při čtení zpráv z QLOCAL.

Protože je upozorněn kterýkoli modul ExceptionListener , který je registrován u ovlivněného připojení, můžete je použít k identifikaci příčiny problému. V některých případech musí administrátor systému zasáhnout a vyřešit problém.

Pro zotavení z těchto chybových stavů použijte jednu z následujících metod:

- Volejte příkaz `close()` na všech ovlivněných ConnectionConsumers. Aplikace může vytvářet nové ConnectionConsumers pouze po uzavření všech ovlivněných ConnectionConsumers a všechny systémové problémy se vyřeší.
- Volejte příkaz `stop()` na všech ovlivněných připojeních. After all Connections are stopped and any system problems are resolved, the application can `start()` its Connections successfully.

Kódy příčiny a zpětné vazby v ASF
Použijte důvod a kódy zpětné vazby k určení příčiny chyby. Zde jsou uvedeny obecné kódy příčiny vygenerované položkou ConnectionConsumer .

Chcete-li určit příčinu chyby, použijte následující informace:

- Kód zpětné vazby ve všech zprávách sestav
- Kód příčiny v MQDLH všech zpráv ve frontě nedoručených zpráv

ConnectionConsumers generují následující kódy příčiny.

MQRC_BACKOUT_THRESHOLD_REACHED (0x93A; 2362)

Příčina
Zpráva dosáhla prahové hodnoty Backout definované na QLOCAL, ale není definována žádná fronta vrácení.

Na platformách, ve kterých nelze definovat frontu vrácení, se zpráva dostala do prahové hodnoty JMSdefinované jako záložní pro 20.

Akce
Pokud to není požadováno, definujte frontu vrácení pro příslušnou QLOCAL. Také se podívejte na příčinu více zákuliů.

MQRC_MSG_NOT_MATCHED (0x93B; 2363)

Příčina
V systému zpráv typu point-to-point je zpráva, která neodpovídá žádnému z selektorů pro monitorování fronty ConnectionConsumers . Chcete-li zachovat výkon, je zpráva znovu zařazena do fronty nedoručených zpráv.

Akce
Chcete-li se této situaci vyhnout, ujistěte se, že ConnectionConsumers používající frontu poskytují sadu selektorů, které se zabývají všemi zprávami, nebo nastavte továrnu QueueConnectionna uchování zpráv.

Případně můžete vyšetřit zdroj zprávy.

MQRC_JMS_FORMAT_ERROR (0x93C; 2364)

Příčina
Produkt JMS nemůže interpretovat zprávu ve frontě.

Akce
Prozkoumejte původ zprávy. Produkt JMS normálně doručuje zprávy neočekávaného formátu jako BytesMessage nebo TextMessage. Občas se to nezdaří, je-li zpráva velmi špatně naformátována.

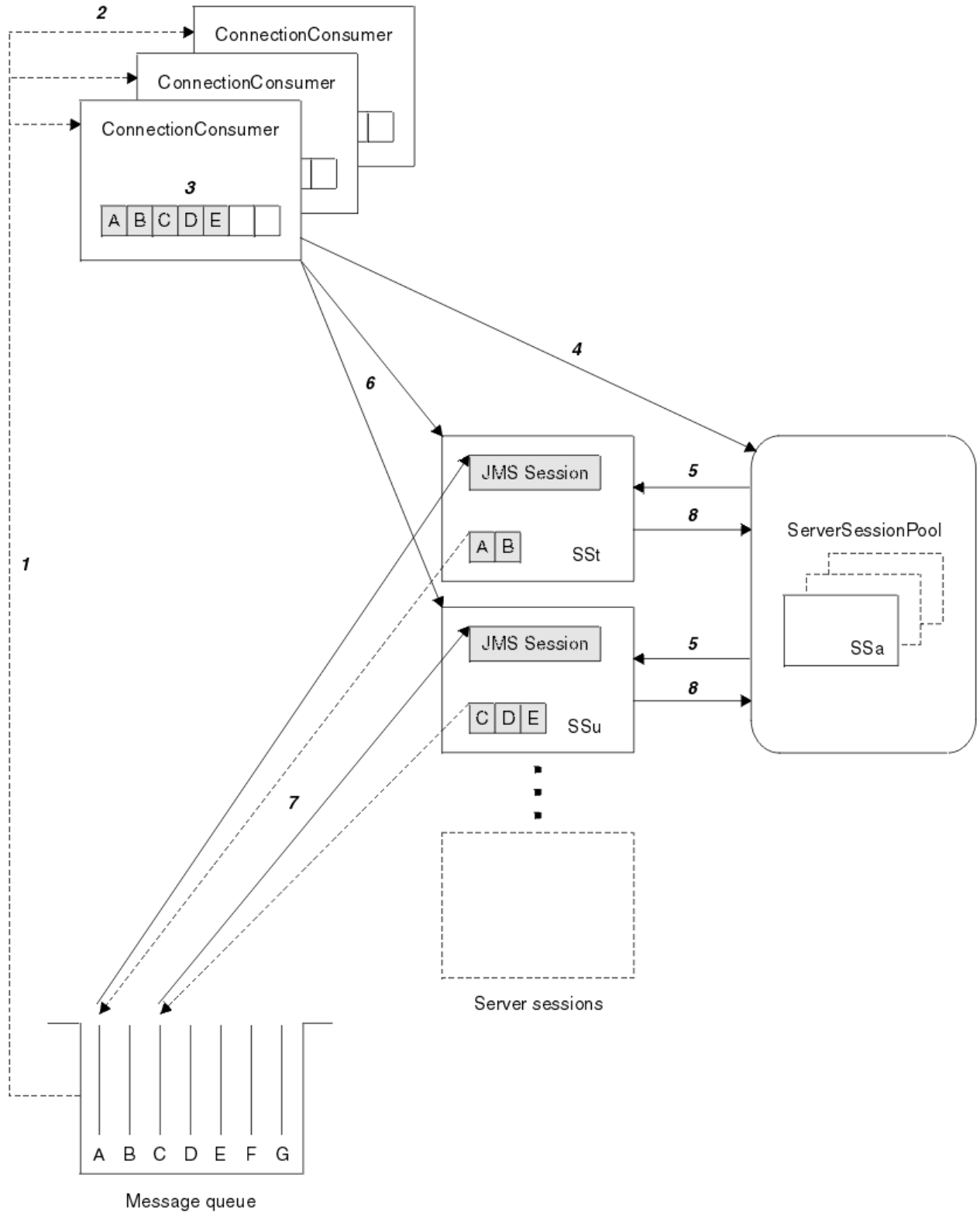
Další kódy, které se objevují v těchto polích, jsou způsobeny neúspěšným pokusem o opětné zařazení zprávy do fronty vrácení. V této situaci kód popisuje příčinu selhání funkce requeue. Chcete-li diagnostikovat příčinu těchto chyb, prohlédněte si téma Kódy dokončení a příčin rozhraní API.

Pokud nelze zprávu sestavy uložit do fronty ReplyTo, vložte ji do fronty nedoručených zpráv. V této situaci je pole zpětné vazby MQMD dokončeno, jak je popsáno v tomto tématu. Pole příčiny v MQDLH vysvětluje, proč nemohla být zpráva sestavy umístěna na ReplyToQ.

Funkce fondu relací serveru v systému AFS

Toto téma shrnuje funkci fondu relací serveru.

Příkaz Obrázek 51 na stránce 301 shrnuje zásady funkcí ServerSessionPool a ServerSession .



Obrázek 51. Funkčnost ServerSessionPool a ServerSession

1. Volba ConnectionConsumers získá odkazy na zprávy z fronty.
2. Každý ConnectionConsumer vybírá specifické odkazy na zprávy.
3. Vyrovnávací paměť ConnectionConsumer obsahuje vybrané odkazy na zprávy.
4. Hodnota ConnectionConsumer vyžaduje jednu nebo více relací ServerSessions ze fondu ServerSession.

5. ServerSessions jsou alokovány ze oblasti ServerSessionPool.
6. Objekt ConnectionConsumer přiřadí odkazy na zprávy do relací ServerSessions a spustí podprocesy ServerSession spuštěné.
7. Každá ServerSession načítá odkazované zprávy z fronty. předává je metodě onMessage z objektu MessageListener , který je přidružen k relaci JMS .
8. Po dokončení zpracování se ServerSession vrátí do fondu.

Aplikační server obvykle poskytuje funkce ServerSessionPool a ServerSession .

z/OS Použití IBM MQ classes for JMS v serveru CICS OSGi JVM

IBM MQ 8.0 added support for using the IBM MQ classes for JMS in certain versions of the CICS Open Services Gateway initiative (OSGi) Java Virtual Machine (JVM) server.

Upozornění: Zkontrolujte systémové požadavky na systém CICS , který váš podnik používá. Podrobné informace naleznete v tématu Podrobné systémové požadavky pro server CICS Transaction Server .

Toto téma představuje úvod k tomu, jak nastavit produkt IBM MQ classes for JMS v prostředí serveru JVM.

Podrobné informace o nastavení a konfiguraci systému viz Použití IBM MQ classes for JMS na serveru prostředí JVM OSGi v dokumentaci produktuCICS .

Obecná omezení

Při používání produktu IBM MQ classes for JMS v serveru CICS OSGi JVM platí následující omezení:

- Připojení v režimu klienta nejsou podporována.
- Připojení jsou podporována pouze pro správce front IBM WebSphere MQ 7.1nebo IBM MQ 8.0 nebo pozdější. Atribut **PROVIDERVERSION** na faktorii připojení musí být buď nespecifikovaný, nebo hodnota větší než nebo rovna sedmi.
- Použití žádné z továren na připojení XA, například `com.ibm.mq.jms.MQXAConnectionFactory`, není podporováno.
- Použití IBM MQ classes for JMS v serveru CICS OSGi JVM je podporováno pouze v produktu CICS 5.2 nebo novějším. Používáte-li produkt CICS 5.2, musíte použít opravu APAR PI32151.

V 9.0.1 Před IBM MQ 9.0.1není podporováno použití IBM MQ classes for JMS v prostředí serveru prostředí JVM produktu Liberty .

Související informace

Konfigurace vlastnosti produktu JMS **PROVIDERVERSION**

V 9.0.1 z/OS Použití prostoru IBM MQ classes for JMS na serveru prostředí JVM produktu CICS Liberty

Z produktu IBM MQ 9.0.1mohou programy Java spuštěné na serveru prostředí CICS Liberty JVM používat server IBM MQ classes for JMS pro přístup k produktu IBM MQ.

Musíte používat IBM MQ 9.0.1 nebo novější verzi IBM MQ, adaptér prostředků, který je možné získat z Fix Central (viz “Instalace adaptéru prostředků v produktu Liberty” na stránce 409).

K dispozici jsou dvě varianty prostředí JVM profilu produktu Liberty dostupné v produktu CICS 5.3 a později jsou typy připojení možných IBM MQ omezeny následujícím způsobem:

CICS Liberty Standardní

- Adaptér prostředků produktu IBM MQ se může připojit k libovolné verzi produktu IBM MQ v režimu CLIENT.

- Adaptér prostředků produktu IBM MQ se může připojit k libovolné verzi produktu IBM MQ for z/OS v režimu BINDINGS, pokud neexistuje žádné připojení produktu CICS (aktivní definice prostředku CICS MQCONN) ke stejnému správci front ze stejné oblasti CICS .

CICS Liberty Integrované

- Adaptér prostředků produktu IBM MQ se může připojit k libovolné verzi IBM MQ v režimu CLIENT.
- Připojení režimu BINDINGS není podporováno.

Další informace naleznete v tématu Použití IBM MQ classes for JMS na serveru Liberty JVM v dokumentaci produktu CICS , kde najdete podrobnosti o nastavení a konfiguraci systému.

Použití IBM MQ classes for JMS v IMS

Produkt IBM MQ 8.0.0 Fix Pack 4 přidal podporu pro použití produktu IBM MQ classes for JMS v produktu IMS verze 13 a novější.

Zkontrolujte systémové požadavky na systém IMS , který váš podnik používá. Další informace naleznete v tématu Obecné informace o plánování pro produkt IMS 13 .

Tato sada témat popisuje, jak nastavit produkt IBM MQ classes for JMS v prostředí produktu IMS a omezení API, která se používají při použití klasických (JMS 1.1) a zjednodušených (JMS 2.0) rozhraní. Seznam informací specifických pro rozhraní API najdete v tématu “Omezení API produktu JMS” na stránce 307 .

Poznámka: Podobná omezení platí pro starší rozhraní (JMS 1.0.2) specifická pro doménu, ale zde nejsou specificky popsány.

Podporované závislé oblasti IMS

Jsou podporovány následující závislé typy oblastí:

- MPR.
- BMP
- IFP
- JMP (31 bit Java virtual machine only, 64 bit JVM nejsou podporovány)
- JBP (31 bit JVM pouze, 64bitová prostředí JVM nejsou podporována)

Pokud není výslovně uvedeno v následujících tématech, IBM MQ classes for JMS se chová stejně ve všech typech oblastí.

Podporované virtuální počítače Java

IBM MQ classes for JMS vyžaduje Java Platform, Standard Edition 7 (Java SE 7) nebo pozdější.

Jiná omezení

Při používání produktu IBM MQ classes for JMS v prostředí produktu IMS platí následující omezení:

- Připojení v režimu klienta nejsou podporována.
- Připojení jsou podporována pouze pro správce front produktu IBM MQ 8.0 s použitím poskytovatele systému zpráv produktu IBM MQ `Normal`nebo produktu `Version 8` .

  Atribut **`PROVIDERVERSION`** na faktorii připojení musí být buď nespecifikovaný, nebo hodnota větší než nebo rovna sedmi.
- Použití žádné z továren na připojení XA, například `com.ibm.mq.jms.MQXAConnectionFactory`, není podporováno.

Související informace

Definování IBM MQ na IMS

Nastavení adaptéru IMS pro použití s produktem IBM MQ classes for JMS

IBM MQ classes for JMS využívá stejný adaptér IBM MQ-IMS , jak je používán jinými programovacími jazyky. Tento adaptér používá modul ESAF (IMS External Subsystem Attach Facility).

Než začnete

Před provedením následujícího postupu je třeba nakonfigurovat adaptér produktu IMS pro příslušné správce front a řídicí a závislé regiony produktu IMS , jak je popsáno v tématu Nastavení adaptéru IMS.

Upozornění: Tento krok, který popisuje sestavení dynamického stubu, není třeba provádět, pokud nepotřebujete dynamický stub pro jiné účely.

Po konfiguraci adaptéru IMS proveďte následující postup.

Postup

1. Aktualizujte proměnnou LIBPATH ve členu produktu IMS PROCLIB, na který odkazuje parametr ENVIRON ve vaší závislé oblasti JCL (například DFJVMEV) tak, aby zahrnoval nativní knihovny produktu IBM MQ classes for JMS .

   To znamená, že adresář zFS obsahuje `libmqjims.so`. Například DFSJVEV může vypadat jako následující, kde poslední řádek je adresář obsahující nativní knihovny produktu IBM MQ classes for JMS :

```
LIBPATH=>
/java/java71_31/J7.1/bin/j9vm:>
/java/java71_31/J7.1/bin:>
/ims13/dbdc/imsjava/classic/lib:>
/ims13/dbdc/imsjava/lib:>
/mqm/V8R0M0/java/lib
```

2. Přidejte IBM MQ classes for JMS do cesty ke třídě prostředí JVM, kterou používá váš IMS závislý region, tak, že aktualizujete volbu `java.class.path` .

   Proveďte toto podle pokynů v členu DFSJVMMS datové sady IMS PROCLIB.

   Můžete například použít následující text, kde čára zvýrazněné tučně označuje aktualizaci:

```
-Djava.class.path=/ims13/dbdc/imsjava/imsutm.jar:/ims13/dbdc/imsjava/imsudb.jar:
/mqm/V8R0M0/java/lib/com.ibm.mq.allclient.jarLIBPATH_SUFFIX=MQ_INSTALLATION_PATH
```

   Poznámka: Zatímco v adresáři obsahujícím IBM MQ classes for JMSje k dispozici mnoho různých souborů JAR, potřebujete pouze soubor `com.ibm.mq.allclient.jar` .

3. Zastavte a znovu spusťte všechny závislé oblasti na serveru IMS , které budou využívat produkt IBM MQ classes for JMS.

Jak pokračovat dále

Vytvořit a konfigurovat továrny připojení a cíle.

Existují tři možné přístupy pro vytváření instancí implementacích IBM MQ továren připojení a cílů. Podrobnosti viz “Vytvoření a konfigurace továren připojení a cílů v aplikaci IBM MQ classes for JMS” na stránce 178.

Všimněte si, že tyto tři přístupy jsou všechny platné v prostředí IMS .

Související informace

Nastavení adaptéru IMS

Definování IBM MQ na IMS

Transakční chování

Zprávy odeslané a přijaté serverem IBM MQ classes for JMS v prostředí IMS jsou vždy přidruženy k pracovní jednotce IMS (UOW), která je aktivní v aktuální úloze.

Tuto jednotku UOW lze dokončit pouze voláním metod pro potvrzení nebo odvolání v instanci objektu `com.ibm.ims.dli.tm.Transaction` nebo úlohou IMS , která končí obvyklým způsobem v tom případě, že je transakce UOW implicitně potvrzena. Pokud se úloha IMS ukončí nestandardně, jednotka UOW se odvolá.

V důsledku toho jsou hodnoty argumentů **`transacted`** a **`acknowledgeMode`** ignorovány při volání jakékoli z metod `Connection.createSession` nebo `ConnectionFactory.createContext`. Navíc nejsou podporovány dále uvedené metody. Volání libovolné z dále uvedených metod způsobí výjimku `IllegalStateException` v případě relace:

- `javax.jms.Session.commit()`
- `javax.jms.Session.recover()`
- `javax.jms.Session.rollback()`

a `IllegalStateRuntimeSession` v případě kontextu JMS:

- `javax.jms.JMSContext.commit()`
- `javax.jms.JMSContext.recover()`
- `javax.jms.JMSContext.rollback()`

K tomuto chování je jedna výjimka. Je-li kontext relace nebo JMS vytvořen pomocí jednoho z následujících mechanizmů:

- `Connection.createSession(false, Session.AUTO_ACKNOWLEDGE)`
- `Connection.createSession(Session.AUTO_ACKNOWLEDGE)`
- `ConnectionFactory.createContext(JMSContext.AUTO_ACKNOWLEDGE)`

pak chování této relace nebo kontextu produktu JMS je následující:

- Všechny odeslané zprávy se odesílají mimo UOW IMS . To znamená, že budou dostupné na cílovém místě určení okamžitě, nebo po dokončení poskytnutého intervalu zpoždění doručení.
- Všechny netrvalé zprávy budou přijaty mimo jednotku UOW IMS , pokud nebyla vlastnost SYNCPALINTALGETS určena v továrně připojení, která vytvořila relaci nebo kontext JMS .
- Trvalé zprávy budou vždy přijaty uvnitř UOW IMS .

To může být užitečné například v případě, že chcete zapsat zprávu auditu do fronty i v případě, že se jednotka UOW vrátí zpět.

Implikace synchronizačních bodů IMS

Sestavení IBM MQ classes for JMS se provádí na stávající podpoře adaptéru IBM MQ , která využívá ESAF. To znamená, že se použije zdokumentované chování, včetně všech otevřených manipulátorů uzavíraných adaptérem IMS , když se vyskytne synchronizační bod.

Další informace viz “Synchronizační body v aplikacích produktu IMS” na stránce 58.

Pro ilustraci tohoto bodu zvažte následující kód spuštěný v prostředí JMP.
Druhé volání příkazu `mp.send()` má za následek `JMSException` , protože kód `messageQueue.getUnique(inputMessage)` má za následek zavření všech otevřených připojení IBM MQ a manipulátorů objektů.

Podobné chování je zaznamenáno, pokud bylo volání `getUnique()` nahrazeno produktem `Transaction.commit()`, ale ne, pokud byl použit `Transaction.rollback()` .

```
//Create a connection to queue manager MQ21.
MQConnectionFactory cf = new MQConnectionFactory();
cf.setQueueManager("MQ21");

Connection c = cf.createConnection();
Session s = c.createSession();

//Send a message to MQ queue Q1.
Queue q = new MQQueue("Q1");
MessageProducer mp = s.createProducer(q);
```

```
TextMessage m = s.createTextMessage("Hello world!");
mp.send(m);

//Get a message from an IMS message queue. This results in a GU call
//which results in all MQ handles being closed.
Application a = ApplicationFactory.createApplication();
MessageQueue messageQueue = a.getMessageQueue();
IOMessage inputMessage = a.getIOMessage(MESSAGE_CLASS_NAME);
messageQueue.getUnique(inputMessage);

//This attempt to send another message will result in a JMSException containing a
//MQRC_HCONN_ERROR as the connection/handle has been closed.
mp.send(m);
```

Správný kód, který má být použit v tomto scénáři, je následující. V tomto případě je připojení k produktu IBM MQ před voláním `getUnique()` uzavřeno. Připojení a relace jsou poté znovu vytvořeny, aby bylo možné odeslat další zprávu.

```
//Create a connection to queue manager MQ21.
MQConnectionFactory cf = new MQConnectionFactory();
cf.setQueueManager("MQ21");

Connection c = cf.createConnection();
Session s = c.createSession();

//Send a message to MQ queue Q1.
Queue q = new MQQueue("Q1");
MessageProducer mp = s.createProducer(q);
TextMessage m = s.createTextMessage("Hello world!");
mp.send(m);

//Close the connection to MQ, which closes all MQ object handles.
//The send of the message will be committed by the subsequent GU call.
c.close();
c = null;
s = null;
mp = null;

//Get a message from an IMS message queue. This results in a GU call.
Application a = ApplicationFactory.createApplication();
MessageQueue messageQueue = a.getMessageQueue();
IOMessage inputMessage = a.getIOMessage(MESSAGE_CLASS_NAME);
messageQueue.getUnique(inputMessage);

//Re-create the connection to MQ and send another message;
c = cf.createConnection();
s = c.createSession();
mp = s.createProducer(q);
m = s.createTextMessage("Hello world 2!");
mp.send(m);
```

Aspekty použití adaptéru IMS

Je třeba, abyste si byli vědomi následujících omezení. Pro každého správce front můžete mít pouze jeden manipulátor připojení. Při použití produktu JMS a nativního kódu jsou k dispozici důsledky pro interakci s produktem IBM MQ . Pro ověření připojení a autorizaci existují omezení.

Jeden popisovač připojení pro každého správce front

V závislých oblastech produktu IMS je povolen pouze jeden manipulátor připojení v daném okamžiku ke specifickému správci front. Veškeré následné pokusy o připojení ke stejnému správci front znovu použijí existující popisovač.

Zatímco toto chování by nemělo způsobit žádné problémy v aplikaci, která používá pouze produkt IBM MQ classes for JMS, může toto chování způsobit problémy v aplikacích interagují s produktem IBM MQpři použití rozhraní IBM MQ classes for JMS a rozhraní MQI v nativním kódu v jazycích, jako je například COBOL nebo C.

Implikace interakce s produktem IBM MQ při použití produktu JMS i nativního kódu

Problémy se mohou vyskytnout při prokládání kódu Java a nativního kódu, které obě používají funkčnost produktu IBM MQ , a když není připojení k produktu IBM MQ ukončeno před opuštěním nativního kódu Java nebo nativního kódu Java.

Například v následujícím pseudokódu je popisovač připojení ke správci front původně zaveden v kódu Java s použitím modulu IBM MQ classes for JMS. Popisovač připojení je znovu použit v kódu COBOL a zneplatněn voláním MQDISC.

Při příštím použití obslužného programu IBM MQ classes for JMS pro použití manipulátoru připojení `JMSException` s kódem příčiny MQRC_HCONN_ERROR.

```
COBOL code running in message processing region
    Use the Java Native Interface (JNI) to call Java code
       Create MQ connection and session - this creates an MQ connection handle
       Send message to MQ queue
       Store connection and session in static variable
       Return to COBOL code

    MQCONN - picks up MQ connection handle established in Java code
    MQDISC - invalidates connection handle

    Use the Java Native Interface (JNI) to call Java code
       Get session from static variable
       Create a message consumer - fails as connection handle invalidated
```

Existují další podobné vzorce použití, které mohou způsobit chybu MQRC_HCONN_ERROR.

I když je možné sdílet manipulátory připojení produktu IBM MQ mezi nativním kódem a kódem jazyka Java (například předchozí příklad by fungoval, kdyby nebylo volání MQDISC) obecně, nejlepším postupem je zavřít všechny obslužné rutiny připojení před změnou z jazyka Java na nativní kód nebo v opačném směru.

Ověřování a autorizace připojení

Specifikace JMS povoluje, aby bylo při vytváření připojení nebo objektu kontextu JMS zadáno jméno uživatele a heslo pro ověření a autorizaci.

To není podporováno v prostředí produktu IMS . Pokus o vytvoření připojení se zadáním jména uživatele a hesla způsobí, že se vyvolá `JMS Exception` . Pokus o vytvoření kontextu produktu JMS a zadání jména uživatele a hesla způsobí, že se bude hodit `JMSRuntimeException` .

Místo toho musí být použity existující mechanismy pro ověření a autorizaci při připojování k produktu IBM MQ z prostředí produktu IMS .

Další informace viz Nastavení zabezpečení v systému z/OS. Zvláště pak téma ID uživatelů pro kontrolu zabezpečení, které popisuje použitelná ID uživatelů.

Související informace

Nastavení zabezpečení v systému z/OS

Omezení API produktu JMS

Z perspektivy specifikace produktu JMS produkt IBM MQ classes for JMS považuje produkt IMS za vyhovující aplikační server Java EE, který má vždy probíhající transakci JTA.

Například nikdy nemůžete volat `javax.jms.Session.commit()` v systému IMS, protože specifikace JMS uvádí, že ji nelze volat v sadě JEE EJB nebo webový kontejner, zatímco transakce JTA probíhá.

To má za následek následující omezení na rozhraní API produktu JMS , kromě těch, které jsou popsány v části “Transakční chování” na stránce 304.

Omezení klasického rozhraní API

- `javax.jms.Connection.createConnectionConsumer(javax.jms.Destination, String, javax.jms.ServerSessionPool, int)` vždy hodí `JMSException`.
- `javax.jms.Connection.createDurableConnectionConsumer(javax.jms.Topic, String, String, javax.jms.ServerSessionPool, int)` vždy hodí `JMSException`.
- Všechny tři varianty `javax.jms.Connection.createSession` vždy vyvolají výjimku `JMSException`, pokud má připojení již existující relaci aktivní.
- `javax.jms.Connection.createSharedConnectionConsumer(javax.jms.Topic, String, String, javax.jms.ServerSessionPool, int)` vždy hodí `JMSException`.
- `javax.jms.Connection.createSharedDurableConnectionConsumer(javax.jms.Topic, String, String, javax.jms.ServerSessionPool, int)` vždy hodí `JMSException`.
- `javax.jms.Connection.setClientID()` vždy hodí `JMSException`.
- `javax.jms.Connection.setExceptionListener(javax.jms.ExceptionListener)` vždy hodí `JMSException`.
- `javax.jms.Connection.stop()` vždy hodí `JMSException`.
- `javax.jms.MessageConsumer.setMessageListener(javax.jms.MessageListener)` vždy hodí `JMSException`.
- `javax.jms.MessageConsumer.getMessageListener()` vždy hodí `JMSException`.
- `javax.jms.MessageProducer.send(javax.jms.Destination, javax.jms.Message,javax.jms.CompletionListener)` vždy hodí `JMSException`.
- `javax.jms.MessageProducer.send(javax.jms.Destination, javax.jms.Message, int, int, long, javax.jms.CompletionListener)` vždy hodí `JMSException`.
- `javax.jms.MessageProducer.send(javax.jms.Message, int, int, long, javax.jms.CompletionListener)` vždy hodí `JMSException`.
- `javax.jms.MessageProducer.send(javax.jms.Message,javax.jms.CompletionListener)` vždy hodí `JMSException`.
- `javax.jms.Session.run()` vždy hodí `JMSRuntimeException`.
- `javax.jms.Session.setMessageListener(javax.jms.MessageListener)` vždy hodí `JMSException`.
- `javax.jms.Session.getMessageListener()` vždy hodí `JMSException`.

Zjednodušená omezení API

- `javax.jms.JMSContext.createContext(int)` vždy hodí `JMSRuntimeException`.
- `javax.jms.JMSContext.setClientID(String)` vždy hodí `JMSRuntimeException`.
- `javax.jms.JMSContext.setExceptionListener(javax.jms.ExceptionListener)` vždy hodí `JMSRuntimeException`.
- `javax.jms.JMSContext.stop()` vždy hodí `JMSRuntimeException`.
- `javax.jms.JMSProducer.setAsync(javax.jms.CompletionListener)` vždy hodí `JMSRuntimeException`.

použitíIBM MQ classes for Java

Použijte IBM MQ v prostředí Java . IBM MQ classes for Java umožňuje aplikaci Java připojit se k IBM MQ jako klient IBM MQ nebo se připojit přímo ke správci front IBM MQ .

Poznámka:

Produkt IBM MQ classes for Java je funkčně stabilizovaný na úrovni dodávané v produktu IBM MQ 8.0. Další informace viz Stabilizace tříd produktu IBM MQ pro jazyk Java.

IBM MQ classes for Java nejsou v produktu IMSpodporovány.

IBM MQ classes for Java nejsou v produktu WebSphere Application Server Libertypodporovány. Nesmí se používat buď s funkcí systému zpráv produktu IBM MQ Liberty , ani s generickou podporou produktu JCA . Další informace naleznete v tématu Použití rozhraní Java produktu WebSphere MQ v prostředí J2EE/JEE Environments.

IBM MQ classes for Java je jedno ze dvou alternativních rozhraní API, které aplikace Java mohou použít pro přístup k prostředkům produktu IBM MQ . Další rozhraní API je IBM MQ classes for JMS.

V produktu IBM MQ 8.0se produkt IBM MQ classes for Java sestavuje s produktem Java 7.

Běhové prostředí produktu Java 7 podporuje spuštění starších verzí souborů tříd.

IBM MQ classes for Java zapouzdřuje rozhraní MQI (Message Queue Interface), nativní rozhraní API produktu IBM MQ a používá podobný model objektu k rozhraním C++ a .NET do produktu IBM MQ.

Programovatelné volby umožňují produktu IBM MQ classes for Java připojit se k produktu IBM MQ jedním z následujících způsobů:

- V režimu klienta jako IBM MQ MQI client pomocí protokolu Transmission Control Protocol/Internet Protocol (TCP/IP)
- V režimu vazebse připojuje přímo k produktu IBM MQ pomocí rozhraní JNI (Java Native Interface).

Poznámka: Automatické opětovné připojení klienta není podporováno produktem IBM MQ classes for Java.

Připojení v režimu klienta

Aplikace produktu IBM MQ classes for Java se může připojit k libovolnému podporovanému správci front pomocí režimu klienta.

Chcete-li se připojit ke správci front v režimu klienta, může být aplikace IBM MQ classes for Java spuštěna na stejném systému, v němž je spuštěn správce front, nebo na jiném systému. V každém případě se produkt IBM MQ classes for Java připojí ke správci front prostřednictvím protokolu TCP/IP.

Další informace o tom, jak zapisovat aplikace pro použití připojení v režimu klienta, najdete v tématu “Režimy připojení produktu IBM MQ classes for Java” na stránce 332.

Připojení režimu vazeb

Při použití v režimu vázání produkt IBM MQ classes for Java používá rozhraní JNI (Java Native Interface) k volání přímo do existujícího rozhraní API správce front, a nikoli prostřednictvím komunikace prostřednictvím sítě. Ve většině prostředí poskytuje připojení v režimu vazeb lepší výkon pro aplikace IBM MQ classes for Java , než se připojuje v režimu klienta, tím, že se vyhnete nákladům na komunikaci TCP/IP.

Aplikace, které používají produkt IBM MQ classes for Java k připojení v režimu vazeb, musí být spuštěny ve stejném systému jako správce front, ke kterému se připojují.

Běhové prostředí produktu Java , které se používá ke spouštění aplikace produktu IBM MQ classes for Java , musí být konfigurováno k načtení knihoven produktu IBM MQ classes for Java ; další informace viz “IBM MQ classes for Java knihovny” na stránce 318 .

Další informace o tom, jak zapisovat aplikace pro použití připojení v režimu vázání, viz “Režimy připojení produktu IBM MQ classes for Java” na stránce 332.

Související informace

Rozhraní jazyka produktu IBM MQ Java

Trasování aplikací produktu IBM MQ classes for Java

Odstraňování problémů s Java a JMS

použitíIBM MQ classes for JMS

Proč bych měl používat produkt IBM MQ classes for Java?

Aplikace Java může použít buď produkt IBM MQ classes for Java , nebo IBM MQ classes for JMS pro přístup k prostředkům produktu IBM MQ .

Poznámka: Ačkoli jsou existující aplikace, které používají produkt IBM MQ classes for Java , i nadále plně podporovány, by nové aplikace měly používat produkt IBM MQ classes for JMS. Funkce, které byly nedávno přidány do produktu IBM MQ, jako např. asynchronní spotřeba a automatické opětovné připojení, nejsou v produktu IBM MQ classes for Javak dispozici, ale jsou k dispozici v produktu IBM MQ classes for JMS. Další informace viz “Proč bych měl používat produkt IBM MQ classes for JMS?” na stránce 70.

Poznámka: IBM MQ classes for Java jsou funkčně stabilizované na úrovni dodávané v produktu IBM MQ 8.0. Další informace naleznete v tématu Stabilizace tříd produktu IBM MQ pro jazyk Java.

Nezbytné předpoklady pro IBM MQ classes for Java

Chcete-li použít produkt IBM MQ classes for Java, potřebujete některé další softwarové produkty.

Informace o předpokladech pro produkt IBM MQ classes for Javanaleznete na webové stránce Systémové požadavky pro IBM MQ .

Chcete-li vyvíjet aplikace IBM MQ classes for Java , potřebujete sadu Java Development Kit (JDK). Podrobnosti o sad JDK podporovaných vaším operačním systémem lze nalézt v informacích o produktu Systémové požadavky pro IBM MQ .

Chcete-li spustit aplikace produktu IBM MQ classes for Java , potřebujete tyto softwarové komponenty:

- Správce front produktu IBM MQ , pro aplikace, které se připojují ke správci front
- Běhové prostředí produktu Java (JRE) pro každý systém, na kterém spouštíte aplikace. Vhodné prostředí JRE je dodáváno s produktem IBM MQ.
- IBM i Pro IBM i, QShell, což je volba 30 operačního systému
- z/OS Pro z/OS, UNIX and Linux System Services (USS)

Požadujete-li připojení TLS k použití šifrovacích modulů, které byly certifikovány FIPS 140-2, potřebujete poskytovatele prostředí FIPS IBM Java JSSE (IBMJSSEFIPS). Každé prostředí IBM JDK a prostředí JRE ve verzi 1.4.2 nebo novější obsahuje IBMJSSEFIPS.

Adresy Internet Protocol verze 6 (IPv6) můžete použít ve svých aplikacích IBM MQ classes for Java pokud IPv6 je podporovaná vaším virtuálním počítačem Java (JVM) a implementací TCP/IP ve vašem operačním systému.

Spuštění aplikací IBM MQ classes for Java v rámci produktu Java EE

Existují určitá omezení a aspekty návrhu, které je třeba vzít v úvahu před použitím produktu IBM MQ classes for Java v produktu Java EE.

Produkt IBM MQ classes for Java má omezení při použití v prostředí Java Platform, Enterprise Edition (Java EE). Dále je třeba vzít v úvahu další aspekty, které je třeba vzít v úvahu při návrhu, implementaci a správě aplikace produktu IBM MQ classes for Java , která je spuštěna v prostředí produktu Java EE . Tato omezení a pokyny jsou popsány v následujících sekcích.

Omezení transakcí JTA

Jediný podporovaný správce transakcí pro aplikace používající produkt IBM MQ classes for Java je samotný IBM MQ . Ačkoli aplikace pod kontrolou JTA může používat IBM MQ classes for Java, žádná práce prováděná prostřednictvím těchto tříd není řízena pracovními jednotkami JTA. Namísto toho tvoří lokální jednotky práce oddělené od těch, které jsou spravovány aplikačním serverem prostřednictvím rozhraní JTA. Zejména žádné odvolání transakce JTA nevedlo k odvolání žádných odeslaných nebo přijatých zpráv. This restriction applies to application or bean managed transactions and to container managed transactions, and all Java EE containers. Chcete-li provádět práci systému zpráv přímo

s produktem IBM MQ uvnitř koordinovaných transakcí aplikačního serveru, je třeba místo toho použít produkt IBM MQ classes for JMS .

Vytvoření podprocesů

Produkt IBM MQ classes for Java vytváří podprocesy interně pro různé operace. Například při spuštění v režimu BINDINGS pro volání přímo v lokálním správci front jsou volání prováděna v podprocesu worker, který byl vytvořen interně produktem IBM MQ classes for Java. Další podprocesy lze vytvořit interně, například vymazat nepoužívaná připojení z fondu připojení nebo odebrat odběry pro ukončené aplikace typu publikování/odběr.

Některé aplikace produktu Java EE (například ty, které jsou spuštěny v kontejnerech EJB a Web), nesmějí vytvářet nové podprocesy. Místo toho se musí všechny práce provádět na hlavních aplikačních podprocesech spravovaných aplikačním serverem. Když aplikace používají produkt IBM MQ classes for Java, nemusí být aplikační server schopen rozlišovat mezi kódem aplikace a kódem IBM MQ classes for Java , takže dříve popsané podprocesy způsobí, že aplikace nebude kompatibilní se specifikací kontejneru. Produkt IBM MQ classes for JMS tyto specifikace produktu Java EE nerozbijí a lze je tedy místo něj použít.

Bezpečnostní omezení

Zásady zabezpečení implementované aplikačním serverem mohou zabránit určitým operacím, které jsou prováděny rozhraním API produktu IBM MQ classes for Java , jako je například vytváření a provoz nových podprocesů obslužného programu (jak je popsáno v předchozích sekcích).

Aplikační servery jsou například standardně spuštěny se zabezpečením produktu Java a umožňují jeho povolení prostřednictvím některé konfigurace specifické pro aplikační server (některé aplikační servery také umožňují podrobnější konfiguraci zásad používaných v produktu Java Security). Je-li zabezpečení produktu Java povoleno, produkt IBM MQ classes for Java může porušit pravidla podprocesů zabezpečení zásad zabezpečení produktu Java definovaná pro aplikační server a rozhraní API nemusí být schopno vytvořit všechny podprocesy, které potřebuje, aby mohla fungovat. Chcete-li zabránit problémům se správou podprocesů, použití příkazu IBM MQ classes for Java není podporováno v prostředích, ve kterých je povoleno zabezpečení produktu Java .

Aspekty izolace aplikace

Zamýšleným přínosem pro spouštění aplikací v prostředí produktu Java EE je izolace aplikace. Návrh a implementace IBM MQ classes for Java předdatum prostředí Java EE . IBM MQ classes for Java lze použít takovým způsobem, který nepodporuje koncepci izolace aplikace. Konkrétní příklady aspektů v této oblasti zahrnují:

- Použití statického nastavení (prostředí JVM) v rámci třídy MQEnvironment, například:
  - ID uživatele a heslo, které má být použito pro identifikaci a ověření připojení
  - název hostitele, port a kanál použité pro připojení klienta
  - Konfigurace TLS pro zabezpečená připojení klienta

  Úprava kterékoli z vlastností MQEnvironment ve prospěch jedné aplikace ovlivní také jiné aplikace používající stejné vlastnosti. Při spuštění v prostředí s více aplikacemi, jako je například produkt Java EE, musí každá aplikace používat svou vlastní odlišnou konfiguraci prostřednictvím vytvoření objektů MQQueueManager se specifickou sadou vlastností, nikoli jako výchozí nastavení vlastností konfigurovaných v rámci třídy MQEnvironment v rámci celého procesu.
- Třída MQEnvironment zavádí počet statických metod, které se chovají globálně na všech aplikacích používajících produkt IBM MQ classes for Java v rámci stejného procesu prostředí JVM, a neexistuje žádný způsob, jak toto chování potlačit pro konkrétní aplikace. Příklady:
  - konfigurace vlastností TLS, jako např. umístění úložiště klíčů
  - konfigurace uživatelských procedur kanálu klienta
  - povolení nebo zakázání trasování diagnostiky

– správa výchozího fondu připojení používaného k optimalizaci využití připojení ke správcům front

Vyvolání těchto metod ovlivní všechny aplikace spuštěné ve stejném prostředí produktu Java EE .

- Sdružování připojení je povoleno, aby byl optimalizován proces vytváření více připojení ke stejnému správci front. Výchozí správce fondu připojení je celoprocesový a je sdílen více aplikacemi. Změny v konfiguraci fondu připojení, jako je nahrazení výchozího správce připojení pro jednu aplikaci pomocí metody MQEnvironment.setDefaultConnectionManager() proto ovlivní ostatní aplikace spuštěné na stejném aplikačním serveru Java EE.
- TLS je konfigurováno pro aplikace používající IBM MQ classes for Java pomocí vlastností třídy MQEnvironment a vlastností objektu MQQueueManager . Není integrován se spravovanou konfigurací zabezpečení samotného aplikačního serveru. Musíte se ujistit, že jste produkt IBM MQ classes for Java nakonfigurovali tak, aby poskytoval požadovanou úroveň zabezpečení, a nepoužívat konfiguraci aplikačního serveru.

Omezení režimu vazeb

Produkt IBM MQ a produkt WebSphere Application Server lze instalovat na stejném počítači, jako jsou hlavní verze správce front a adaptéru prostředků produktu IBM MQ (RA) dodávané v produktu WebSphere Application Server . Například produkt WebSphere Application Server 7.0, který je dodáván s úrovní IBM MQ RA 7.0.1, může být nainstalován na stejném počítači jako správce front produktu IBM WebSphere MQ 6 .

Pokud se správce front a hlavní verze adaptéru prostředků liší, nelze použít připojení vazeb. Všechna připojení z produktu WebSphere Application Server ke správci front s použitím adaptéru prostředků musí používat připojení typu klienta. Spojení vazeb lze použít, pokud jsou verze stejné.

Konverze řetězcových řetězců v produktu IBM MQ classes for Java

Produkt IBM MQ classes for Java používá CharsetEncoders a CharsetDecoders přímo pro převod znakových řetězců. Výchozí chování pro převod znakového řetězce lze nakonfigurovat se dvěma vlastnostmi systému. Zpracování zpráv, které obsahují nemapovatelné znaky, lze konfigurovat prostřednictvím `com.ibm.mq.MQMD`.

Před IBM MQ 8.0byla konverze řetězců v IBM MQ classes for Java provedena voláním metod `java.nio.charset.Charset.decode(ByteBuffer)` a `Charset.encode(CharBuffer)` .

Použití jedné z těchto metod vede k výchozí náhradě (REPLACE) chybných nebo nepřeložitelných dat. Toto chování může zakrývat chyby v aplikacích a může vést k neočekávaným znakům, například ?, v přeložených datech.

Z produktu IBM MQ 8.0je možné přímo a efektivněji detekovat tyto problémy IBM MQ classes for Java pomocí CharsetEncoders a CharsetDecoders přímo a explicitně konfigurovat zpracování deformovaných a nepřeložitelných dat. Výchozí chování je takové problémy REPORT , které vyvolává vhodné `MQException`.

Konfigurace

Překládání z UTF-16 (znakové znázornění použité v Java) na nativní znakovou sadu, jako je UTF-8, se označuje jako `encoding`, zatímco překlad v opačném směru se označuje jako `decoding`.

V současné době dekóduje výchozí chování produktu `CharsetDecoders`a hlásí chyby vyvoláním výjimky.

Jedno nastavení se používá k uvedení `java.nio.charset.CodingErrorAction` pro řízení zpracování chyb při kódování i dekódování. Jedno další nastavení se používá k řízení náhradního bajtu, nebo bajtů, při kódování. Výchozí řetězec náhrady Java bude použit při dekódování operací.

Konfigurace zpracování nepřeložitelných dat v produktu IBM MQ classes for Java

V produktu IBM MQ 8.0zahrnuje produkt `com.ibm.mq.MQMD` následující dvě pole:

byte [] unMappablenáhrady
Posloupnost bajtů, která bude zapsána do kódovaného řetězce, pokud vstupní znak nelze přeložit a zadali jste REPLACE.

Výchozí: "?".getBytes()
Výchozí řetězec náhrady Java se používá při dekódování operací.

java.nio.charset.CodingErrorAction unmappableAction
Určuje akci, která má být provedena pro nepřeložitelná data při kódování a dekódování:

Výchozí: CodingErrorAction.REPORT;

Systémové vlastnosti pro nastavení výchozích hodnot systému

V produktu IBM MQ 8.0jsou k dispozici následující dvě vlastnosti systému produktu Java , které slouží ke konfiguraci výchozího chování při převodu znakových řetězců.

com.ibm.mq.cfg.jmqi.UnmappableCharacterAction
Určuje akci, která má být provedena pro nepřeložitelná data při kódování a dekódování. Hodnota může být REPORT, REPLACEnebo IGNORE.

com.ibm.mq.cfg.jmqi.UnmappableCharacterReplacement
Nastaví nebo získává náhradní bajty, aby se aplikoval, když nelze namapovat znak v operaci kódování Výchozí řetězec náhrady Java se používá při dekódování operací.

Chcete-li se vyhnout nejasnostem mezi znakem Java a rodilými bajtovými reprezentacemi, měli byste uvést `com.ibm.mq.cfg.jmqi.UnmappableCharacterReplacement` jako desítkové číslo představující nahrazovací byt v nativní znakové sadě.

Například dekadická hodnota ?jako nativní bajt je 63, pokud je nativní znaková sada založená na ASCII, jako je ISO-8859-1, zatímco je nativní znaková sada 111, je-li nativní znaková sada EBCDIC.

Poznámka: Všimněte si, že pokud má objekt MQMD nebo MQMessage buď sadu polí **`unmappableAction`** , nebo **`unMappableReplacement`** , pak hodnoty těchto polí mají přednost před vlastnostmi systému Java . To umožňuje přepsat hodnoty zadané ve vlastnostech systému Java pro každou zprávu, je-li to nutné.

Související pojmy
“Konverze řetězcových řetězců v produktu IBM MQ classes for JMS” na stránce 116
Produkt IBM MQ classes for JMS používá CharsetEncoders a CharsetDecoders přímo pro převod znakových řetězců. Výchozí chování pro převod znakového řetězce lze nakonfigurovat se dvěma vlastnostmi systému. Zpracování zpráv obsahujících nemapovatelné znaky lze nakonfigurovat prostřednictvím vlastností zprávy pro nastavení akce UnmappableCharactera bajtů náhrady.

Instalace a konfigurace produktu IBM MQ classes for Java

Tento oddíl popisuje adresáře a soubory, které jsou vytvořeny při instalaci produktu IBM MQ classes for Java, a dozvíte se, jak nakonfigurovat produkt IBM MQ classes for Java po instalaci.

Co je nainstalováno pro IBM MQ classes for Java

Nejnovější verze produktu IBM MQ classes for Java je nainstalována s produktem IBM MQ. Je možné, že budete muset přepsat výchozí volby instalace, abyste se ujistili, že je to hotovo.

Další informace o instalaci produktu IBM MQ naleznete v následujících tématech:

- Multi Instalace produktu IBM MQ
- z/OS Instalace produktu IBM MQ for z/OS

IBM MQ classes for Java jsou obsaženy v souborech archivu Java (JAR), `com.ibm.mq.jara` `com.ibm.mq.jmqi.jar`.

Podpora pro standardní záhlaví zpráv, jako např. Programmable Command Format (PCF), je obsažena v souboru JAR `com.ibm.mq.headers.jar`.

Podpora programu Programmable Command Format (PCF) je obsažena v souboru JAR `com.ibm.mq.pcf.jar`.

Poznámka: Nedoporučuje se používat IBM MQ classes for Java v rámci aplikačního serveru. Další informace o omezeních platných při spuštění v tomto prostředí viz “Spuštění aplikací IBM MQ classes for Java v rámci produktu Java EE” na stránce 310. Další informace naleznete v tématu Použití rozhraní Java produktu WebSphere MQ v prostředí J2EE/JEE Environments.

Důležité: Kromě přemístitelných souborů JAR popsaných v tomto tématu není podporováno kopírování souborů JAR produktu IBM MQ classes for Java nebo nativních knihoven do jiných počítačů nebo do jiného umístění na počítači, na kterém byl nainstalován produkt IBM MQ classes for Java . Kromě toho není podporováno, včetně souboru `com.ibm.mq.allclient.jar` nebo IBM MQ classes for Java, v archivech aplikace (jako jsou například archivy podnikových aplikací nebo soubory EAR).

V 9.0.0.3 V 9.0.5 Soubor `JSON4J.jar` a balík `com.ibm.msg.client.mqlight` nejsou vyžadovány IBM MQ classes for Java a IBM MQ classes for JMS. V produktu IBM MQ 9.0.0 Fix Pack 3 a IBM MQ 9.0.5jsou proto provedeny následující změny souboru `com.ibm.mq.allclient.jar` :

- Odkaz na soubor `JSON4J.jar` se odstraní ze souboru cesty ke třídě v souboru typu manifest pro soubor `com.ibm.mq.allclient.jar` .
- Balík `com.ibm.msg.client.mqlight` již není zahrnut do souboru `com.ibm.mq.allclient.jar` .

Přeložitelné soubory JAR

V rámci podniku lze následující soubory přesunout do systémů, které potřebují spustit aplikace produktu IBM MQ classes for Java :

- `com.ibm.mq.allclient.jar`
- `com.ibm.mq.traceControl.jar`
- Poskytovatel zabezpečení Bouncy Castle Security a soubory JAR podpory CMS

Je požadován poskytovatel zabezpečení Bouncy Castle Security a soubory JAR podpory CMS. Další informace naleznete v tématu Podpora pro jiná prostředí nežIBM JRE. Požadují se následující soubory JAR:

- `bcpkix-jdk15on.jar`
- `bcprov-jdk15on.jar`
- V 9.0.0.12 `bcutil-jdk15on.jar`

Soubor `com.ibm.mq.allclient.jar` obsahuje IBM MQ classes for JMS, IBM MQ classes for Javaa třídy PCF a Headers. Přesunete-li tento soubor do nového umístění, ujistěte se, že jste provedli kroky k zachování tohoto nového umístění s novými opravami Fix Pack produktu IBM MQ . Také se ujistěte, že je použití tohoto souboru známo podpoře produktu IBM , pokud máte prozatímní opravu.

Chcete-li určit verzi souboru `com.ibm.mq.allclient.jar` , použijte příkaz:

```
java -jar com.ibm.mq.allclient.jar
```

Následující příklad zobrazuje ukázkový výstup z tohoto příkazu:

```
C:\Program Files\IBM\MQ_1\java\lib>java -jar com.ibm.mq.allclient.jar
Name:       Java Message Service Client
Version:    9.0.0.0
Level:      p000-L140428.1
Build Type: Production
Location:   file:/C: /Program Files/IBM/MQ_1/java/lib/com.ibm.mq.allclient.jar

Name:       WebSphere MQ classes for Java Message Service
Version:    9.0.0.0
Level:      p000-L140428.1
Build Type: Production
Location:   file:/C: /Program Files/IBM/MQ_1/java/lib/com.ibm.mq.allclient.jar
```

```
Name:       WebSphere MQ JMS Provider
Version:    9.0.0.0
Level:      p000-L140428.1 mqjbnd=p000-L140428.1
Build Type: Production
Location:   file:/C: /Program Files/IBM/MQ_1/java/lib/com.ibm.mq.allclient.jar

Name:       Common Services for Java Platform, Standard Edition
Version:    9.0.0.0
Level:      p000-L140428.1
Build Type: Production
Location:   file:/C: /Program Files/IBM/MQ_1/java/lib/com.ibm.mq.allclient.jar
```

Soubor `com.ibm.mq.traceControl.jar` se používá k dynamickému řízení trasování pro aplikace produktu IBM MQ classes for JMS . Další informace viz Řízení trasování ve spuštěném procesu pomocí tříd produktu IBM MQ pro třídy Java a IBM MQ pro platformu JMS.

Instalační adresáře produktu IBM MQ classes for Java
Soubory a ukázky produktu IBM MQ classes for Java jsou instalovány v různých umístěních podle platformy. Umístění prostředí JRE (Java Runtime Environment), které je nainstalováno s produktem IBM MQ , se také liší podle platformy.

Instalační adresáře pro soubory produktu IBM MQ classes for Java

Tabulka 49 na stránce 315 zobrazuje, kde jsou instalovány soubory IBM MQ classes for Java .

Tabulka 49. Instalační adresáře produktu IBM MQ classes for Java

Platforma	Adresář
AIX AIX	*MQ_INSTALLATION_PATH*/java/bin
Solaris Linux HP-UX Solaris Linux HP-UX HP-UX, Linuxa Solaris	*MQ_INSTALLATION_PATH*/java/bin
IBM i IBM i	/QIBM/ProdData/mqm/java/lib
Windows Windows	*MQ_INSTALLATION_PATH*\java\lib
z/OS z/OS	*MQ_INSTALLATION_PATH*/mqm/V8R0M0/java /lib

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Instalační adresáře pro ukázky

Některé ukázkové aplikace, jako např. Ověřovací programy instalace (IVP), jsou dodávány s produktem IBM MQ. Tabulka 50 na stránce 315 zobrazuje, kde jsou nainstalovány ukázkové aplikace. Ukázky IBM MQ classes for Java se nacházejí v podadresáři s názvem `wmqjava`. Ukázky PCF se nacházejí v podadresáři s názvem `pcf`.

Tabulka 50. Adresáře ukázek

Platforma	Adresář
AIX AIX	*MQ_INSTALLATION_PATH*/samp/wmqjava/

Tabulka 50. Adresáře ukázek (pokračování)	
Platforma	**Adresář**
Solaris Linux HP-UX Solaris Linux HP-UX HP-UX, Linuxa Solaris	*MQ_INSTALLATION_PATH*/samp/wmqjava/
IBM i IBM i IBM i	/QIBM/ProdData/mqm/java/samples
Windows Windows Windows	*MQ_INSTALLATION_PATH*\tools\wmqjava\
z/OS z/OS z/OS	*MQ_INSTALLATION_PATH*/mqm/V8R0M0/java/samples

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Instalační adresáře pro prostředí JRE

The IBM MQ classes for JMS require a Java 7 (or above) Java Runtime Environment (JRE). Vhodné prostředí JRE je nainstalováno s produktem IBM MQ. Tabulka 51 na stránce 316 zobrazuje, kde je toto prostředí JRE nainstalované. Chcete-li spustit programy Java , jako jsou např. dodané ukázky, s použitím tohoto prostředí JRE, buď explicitně vyvolejte produkt *JRE_LOCATION*`/bin/java` , nebo přidejte *JRE_LOCATION*`/bin` do prostředí PATH (nebo ekvivalentní) pro vaši platformu, kde *JRE_LOCATION* je adresář poskytnutý v produktu Tabulka 51 na stránce 316.

Tabulka 51. Adresáře JRE	
Platforma	**Adresář**
AIX AIX AIX	*MQ_INSTALLATION_PATH*/java/j
Solaris Linux HP-UX Solaris Linux HP-UX HP-UX, Linuxa Solaris	*MQ_INSTALLATION_PATH*/java/j
IBM i IBM i IBM i	/QIBM/ProdData/mqm/java/jre
Windows Windows Windows	*MQ_INSTALLATION_PATH*\java\jre.
z/OS z/OS z/OS	*MQ_INSTALLATION_PATH*/mqm/V8R0M0/java/jre

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Proměnné prostředí relevantní pro produkt IBM MQ classes for Java
Chcete-li spustit aplikace produktu IBM MQ classes for Java , musí cesta ke třídám obsahovat adresáře IBM MQ classes for Java a ukázky.

Aby bylo možné spustit aplikace IBM MQ classes for Java, musí cesta ke třídám obsahovat příslušný adresář IBM MQ classes for Java . Chcete-li spustit ukázkové aplikace, musí cesta ke třídám obsahovat také příslušné adresáře ukázek. Tyto informace mohou být poskytnuty v příkazu vyvolání Java nebo v proměnné prostředí CLASSPATH.

Důležité: Nastavení volby jazyka Java *-Xbootclasspath*tak, aby zahrnovalo IBM MQ classes for Java, není podporováno.

Tabulka 52 na stránce 317 zobrazuje příslušnou hodnotu CLASSPATH, která má být použita na každé platformě ke spuštění aplikací produktu IBM MQ classes for Java , včetně ukázkových aplikací.

Tabulka 52. CLASSPATH nastavení ke spuštění aplikací produktu IBM MQ classes for Java , včetně ukázkových aplikací IBM MQ classes for Java

Platforma	Nastavení CLASSPATH
AIX AIX AIX	CLASSPATH= *MQ_INSTALLATION_PATH*/java/lib/com.ibm.mq.jar: *MQ_INSTALLATION_PATH*/samp/wmqjava/samples:
Solaris Linux HP-UX Solaris Linux HP-UX HP-UX, Linuxa Solaris	CLASSPATH= *MQ_INSTALLATION_PATH*/java/lib/com.ibm.mq.jar: *MQ_INSTALLATION_PATH*/samp/wmqjava/samples:
IBM i IBM i IBM i	CLASSPATH=/QIBM/ProdData/mqm/java/lib/com.ibm.mq.jar: /QIBM/ProdData/mqm/java/samples/wmqjava/samples:
Windows Windows Windows	CLASSPATH= *MQ_INSTALLATION_PATH*\Java\lib\com.ibm.mq.jar; *MQ_INSTALLATION_PATH*\tools\wmqjava\samples;
z/OS z/OS z/OS	CLASSPATH= *MQ_INSTALLATION_PATH*/mqm/V8R0M0/java/lib/com.ibm.mq.jar: *MQ_INSTALLATION_PATH*/mqm/V8R0M0/java/samples/wmqjava: *MQ_INSTALLATION_PATH*/mqm/V8R0M0/java/samples/pcf

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Při kompilaci pomocí volby -Xlint se může zobrazit zpráva s varováním, že soubor com.ibm.mq.ese.jar není přítomen. Varování můžete ignorovat. Tento soubor je k dispozici pouze v případě, že jste nainstalovali produkt Advanced Message Security.

Skripty poskytnuté s produktem IBM MQ classes for JMS používají následující proměnné prostředí:

MQ_JAVA_DATA_PATH
Tato proměnná prostředí určuje adresář pro výstup protokolu a trasování.

INSTALAČNÍ_CESTA MQ_JAVA_INSTALL_PATH
Tato proměnná prostředí určuje adresář, kde je nainstalován produkt IBM MQ classes for Java , jak je zobrazeno v instalačních adresářích produktu IBM MQ classes for Java.

KOŘEN ROZHRANÍ MQ_JAVA_LIB_PATH
Tato proměnná prostředí určuje adresář, kde jsou uloženy knihovny produktu IBM MQ classes for Java , jak ukazuje Umístění knihoven produktu IBM MQ classes for Java pro každou platformu. Některé skripty dodávané s produktem IBM MQ classes for Java, např. IVTRun, používají tuto proměnnou prostředí.

Windows V systému Windowsjsou všechny proměnné prostředí nastaveny automaticky během instalace.

UNIX V systému UNIXmůžete použít skript **setjmsenv** (používáte-li 32bitové prostředí JVM) nebo **setjmsenv64** (pokud používáte 64bitové prostředí JVM), abyste nastavili proměnné prostředí.

Linux UNIXV systému UNIX and Linuxse tyto skripty nacházejí v adresáři *MQ_INSTALLATION_PATH*/java/bin .

IBM iV systému IBM imusí být proměnná prostředí QIBM_MULTI_THREADED nastavena na hodnotu Y. Poté můžete spustit více aplikací s podporou podprocesů stejným způsobem, jakým spouštíte jednotlivé aplikace s podporou podprocesů. Další informace naleznete v tématu Nastavení produktu IBM MQ pomocí jazyka Java a platformy JMS .

Produkt IBM MQ classes for Java vyžaduje prostředí Java 7 Java Runtime Environment (JRE). Informace o umístění vhodného prostředí JRE, které je nainstalováno s produktem IBM MQ, viz “Instalační adresáře produktu IBM MQ classes for Java” na stránce 315.

IBM MQ classes for Java knihovny
Umístění knihoven IBM MQ classes for Java se liší v závislosti na platformě. Uveďte toto umístění, když spustíte aplikaci.

Chcete-li zadat umístění knihoven JNI (Java Native Interface), spusťte aplikaci pomocí příkazu **java** s následujícím formátem:

```
java -Djava.library.path= library_path application_name
```

kde *cesta_knihovny* je cesta k serveru IBM MQ classes for Java, který zahrnuje knihovny JNI. Tabulka 53 na stránce 318 zobrazuje umístění knihoven produktu IBM MQ classes for Java pro každou platformu. V této tabulce znázorňuje *MQ_INSTALLATION_PATH* vysokoúrovňový adresář, ve kterém je nainstalován produkt IBM MQ .

Tabulka 53. Umístění knihoven produktu IBM MQ classes for Java pro každou platformu

Platforma	**Adresář obsahující knihovny IBM MQ classes for Java**
AIX	*MQ_INSTALLATION_PATH*/java/lib (32bitové knihovny) *MQ_INSTALLATION_PATH*/java/lib64 (64bitové knihovny)
HP-UX Linux (POWER, x86-64 a zSeries s390x platformy) Solaris (platformyx86-64 a SPARC)	*MQ_INSTALLATION_PATH*/java/lib (32bitové knihovny) *MQ_INSTALLATION_PATH*/java/lib64 (64bitové knihovny)
Linux (platformax86)	*MQ_INSTALLATION_PATH*/java/bin
Windows	*MQ_INSTALLATION_PATH*\Java\lib (32bitové knihovny) *MQ_INSTALLATION_PATH*\Java\lib64 (64-bitové knihovny)
z/OS z/OSz/OS	*MQ_INSTALLATION_PATH*/mqm/V8R0M0/java/lib (32bitové a 64bitové knihovny)

Poznámka:

1. Solaris Linux HP-UX AIXV systémech AIX, HP-UX, Linux (Platforma Power) nebo Solarispoužijte 32bitové knihovny nebo 64bitové knihovny. 64bitové knihovny používejte

pouze v případě, že spouštíte aplikaci v 64bitovém prostředí JVM (Java virtual machine) na 64bitové platformě. Jinak použijte 32bitovou knihovnu.

2. Windows V systému Windowsmůžete pomocí proměnné prostředí PATH zadat umístění knihoven produktu IBM MQ classes for Java místo zadávání jejich umístění v příkazu **java** .
3. IBM i Chcete-li produkt IBM MQ classes for Java používat v režimu vazeb v systému IBM i, ujistěte se, že knihovna QMQMJAVA je ve vašem seznamu knihoven.
4. z/OS V systému z/OSmůžete použít buď 32bitový, nebo 64bitový virtuální počítač Java (JVM). Nemusíte určovat, které knihovny použít; IBM MQ classes for Java může určit, které knihovny JNI se mají načíst.

Související pojmy

použitíIBM MQ classes for Java
Po instalaci produktu IBM MQ classes for Javamůžete nakonfigurovat svou instalaci tak, abyste spouštěli vlastní aplikace.

Podpora pro OSGi s produktem IBM MQ classes for Java
OSGi poskytuje rámec, který podporuje implementaci aplikací jako balíky. Tři balíky OSGi jsou dodávány jako součást produktu IBM MQ classes for Java.

Specifikace OSGi poskytuje obecný účel, zabezpečený a spravovaný rámec Java , který podporuje implementaci aplikací, které jsou dodávány ve formě balíků. Zařízení vyhovující specifikaci OSGi mohou stahovat a instalovat balíky a odebírat je, pokud již nejsou zapotřebí. Tento rámec spravuje instalaci a aktualizaci balíků v dynamickém a přizpůsobitelném stylu.

Produkt IBM MQ classes for Java obsahuje následující svazky balíků OSGi.

com.ibm.mq.osgi.java_*version_number*.jar
Soubory JAR, které umožní aplikacím používat produkt IBM MQ classes for Java.

com.ibm.mq.osgi.allclient_*version_number*.jar
Tento soubor JAR umožňuje aplikacím používat jak IBM MQ classes for JMS , tak i kód IBM MQ classes for Java, a také obsahuje kód pro zpracování zpráv PCF.

com.ibm.mq.osgi.allclientprereqs_*version_number*.jar
Tento soubor JAR poskytuje předpoklady pro produkt `com.ibm.mq.osgi.allclient_version_number.jar`.

kde *version_number* je číslo verze produktu IBM MQ , který byl nainstalován.

Balíky jsou nainstalované do podadresáře `java/lib/OSGi` vaší instalace produktu IBM MQ nebo složky `java\lib\OSGi` v systému Windows.

V produktu IBM MQ 8.0použijte balíky `com.ibm.mq.osgi.allclient_8.0.0.0.jara com.ibm.mq.osgi.allclientprereqs_8.0.0.0.jar` pro všechny nové aplikace. Použití těchto balíků odebere omezení, že nebude možné spustit jak produkt IBM MQ classes for JMS , tak produkt IBM MQ classes for Java v rámci stejného rámce OSGi. Všechna ostatní omezení však platí i nadále. U verzí starších než IBM MQ 8.0se používá omezení použití produktu IBM MQ classes for JMS nebo IBM MQ classes for Java .

Devět dalších balíků se nainstaluje také do podadresáře `java/lib/OSGi` instalace produktu IBM MQ nebo do složky `java\lib\OSGi` v systému Windows. Tyto balíky jsou součástí produktu IBM MQ classes for JMSa nesmí být načteny do běhového prostředí OSGi, které má načtený balík IBM MQ classes for Java . Je-li balík OSGi produktu IBM MQ classes for Java načten do běhového prostředí OSGi, které má také načtené svazky balíků produktu IBM MQ classes for JMS , dojde k chybám uvedeným v následujícím příkladu při spuštění aplikací s použitím svazku balíků produktu IBM MQ classes for Java nebo balíků produktu IBM MQ classes for JMS :

```
java.lang.ClassCastException: com.ibm.mq.MQException incompatible with com.ibm.mq.MQException
```

Svazek balíků OSGi pro produkt IBM MQ classes for Java byl zapsán do specifikace OSGi Release 4. Nepracuje v prostředí OSGi Release 3.

Musíte správně nastavit systémovou cestu nebo cestu ke knihovně tak, aby běhové prostředí OSGi mohlo najít všechny požadované soubory DLL nebo sdílené knihovny.

Pokud použijete svazek balíků OSGi pro produkt IBM MQ classes for Java, třídy uživatelské procedury kanálu zapsané v produktu Java nejsou podporovány z důvodu inherentní chyby při načítání tříd v prostředí zavaděče více tříd, jako je např. OSGi. Balík uživatelů si může být vědom balíku IBM MQ classes for Java , ale balík IBM MQ classes for Java není informován o žádném uživatelském balíku. V důsledku toho zavaděč tříd použitý v balíku IBM MQ classes for Java nemůže načíst třídu ukončení kanálu, která se nachází v uživatelském balíku.

Další informace o specifikaci OSGi naleznete na webu Alianci OSGi .

z/OS *Instalace produktu IBM MQ classes for Java v systému z/OS*

V systému z/OSmusí STEPLIB použité v běhovém prostředí obsahovat knihovny IBM MQ SCSQAUTH a SCSQANLE.

V produktu UNIX and Linux System Services můžete tyto knihovny přidat pomocí čáry ve vaší `.profile` , jak je zobrazeno v následujícím příkladu, nahrazením `thlqual` s kvalifikátorem datové sady vysoké úrovně, který jste zvolili při instalaci produktu IBM MQ:

```
export STEPLIB=thlqual.SCSQAUTH:thlqual.SCSQANLE:$STEPLIB
```

V jiných prostředích je zpravidla třeba upravit spouštěcí skript JCL, aby obsahoval SCSQAUTH ve zřetězení STEPLIB:

```
 STEPLIB DD DSN=thlqual.SCSQAUTH,DISP=SHR
         DD DSN=thlqual.SCSQANLE,DISP=SHR
```

Konfigurační soubor IBM MQ classes for Java

Konfigurační soubor IBM MQ classes for Java určuje vlastnosti, které se používají ke konfiguraci produktu IBM MQ classes for Java.

Formát konfiguračního souboru IBM MQ classes for Java je standardní soubor vlastností Java .

V 9.0.0.2 V 9.0.3 V adresáři IBM MQ 9.0.3 a IBM MQ 9.0.0 Fix Pack 2se v podadresáři `bin` instalačního adresáře produktu IBM MQ classes for Java dodává ukázkový konfigurační soubor `mqjava.config`. Tento soubor dokumentuje všechny podporované vlastnosti a jejich výchozí hodnoty.

Poznámka: Ukázkový konfigurační soubor je přepsán při upgradu instalace produktu IBM MQ na budoucí opravnou sadu Fix Pack. Proto se doporučuje vytvořit kopii ukázkového konfiguračního souboru pro použití s vašimi aplikacemi.

Můžete zvolit název a umístění konfiguračního souboru IBM MQ classes for Java . Když spustíte aplikaci, použijte příkaz **java** s následujícím formátem:

```
java -Dcom.ibm.msg.client.config.location=config_file_url application_name
```

V příkazu *config_file_url* je adresa URL (Uniform Resource Locator), která určuje název a umístění konfiguračního souboru IBM MQ classes for Java . Podporovány jsou adresy URL následujících typů: `http`, `file`, `ftp`a `jar`.

Následující příklad ukazuje příkaz **java** :

```
java -Dcom.ibm.msg.client.config.location=file:/D:/mydir/mqjava.config MyAppClass
```

Tento příkaz identifikuje konfigurační soubor IBM MQ classes for Java jako soubor `D:\mydir\mqjava.config` v lokálním systému Windows .

Když se spustí aplikace, produkt IBM MQ classes for Java načte obsah konfiguračního souboru a uloží zadané vlastnosti do interního úložiště vlastností. Pokud příkaz **java** neidentifikuje konfigurační soubor,

nebo pokud nelze konfigurační soubor nalézt, použije produkt IBM MQ classes for Java výchozí hodnoty pro všechny vlastnosti. V případě potřeby můžete přepsat jakoukoli vlastnost v konfiguračním souboru tak, že ji uvedete jako systémovou vlastnost v příkazu **java** .

Konfigurační soubor produktu IBM MQ classes for Java lze použít s libovolným z podporovaných přenosů mezi aplikací a správcem front nebo zprostředkovatelem.

Potlačení vlastností zadaných v konfiguračním souboru IBM MQ MQI client

Konfigurační soubor IBM MQ MQI client může také určovat vlastnosti, které se používají ke konfiguraci produktu IBM MQ classes for Java. Vlastnosti zadané v konfiguračním souboru produktu IBM MQ MQI client se však používají pouze v případě, že se aplikace připojuje ke správci front v režimu klienta.

Je-li to požadováno, můžete přepsat jakýkoli atribut v konfiguračním souboru IBM MQ MQI client tak, že jej uvedete jako vlastnost do konfiguračního souboru IBM MQ classes for Java . Chcete-li přepsat atribut v konfiguračním souboru IBM MQ MQI client , použijte záznam s následujícím formátem v konfiguračním souboru IBM MQ classes for Java :

```
com.ibm.mq.cfg.stanza.propName=propValue
```

Proměnné v položce mají následující význam:

sekce
Název stanzy v konfiguračním souboru IBM MQ MQI client , který obsahuje atribut.

propName
Název atributu, jak je uveden v konfiguračním souboru IBM MQ MQI client .

propValue
Hodnota vlastnosti, která přepíše hodnotu atributu, která je uvedena v konfiguračním souboru IBM MQ MQI client .

Eventuálně můžete přepsat atribut v konfiguračním souboru IBM MQ MQI client zadáním vlastnosti jako systémové vlastnosti v příkazu **java** . Chcete-li určit vlastnost jako systémovou vlastnost, použijte předchozí formát.

Pouze následující atributy v konfiguračním souboru IBM MQ MQI client jsou relevantní pro IBM MQ classes for Java. Pokud uvedete nebo přepisují jiné atributy, nemá žádný efekt. Konkrétně si všimněte, že `ChannelDefinitionFile` a `ChannelDefinitionDirectory` v sekci CHANNELS v konfiguračním souboru klienta se nepoužívají. Podrobné informace o použití tabulky CCDT s produktem IBM MQ classes for Javanaleznete v příručce “Použití tabulky definic kanálů klienta s IBM MQ classes for Java” na stránce 336 .

Tabulka 54. Který oddíl konfiguračního souboru klienta obsahuje který atribut

sekce	**Atribut**
stanza CHANNELS konfiguračního souboru klienta	Put1DefaultAlwaysSync
stanza CHANNELS konfiguračního souboru klienta	PasswordProtection
ClientExitSekce Cesta ke konfiguračnímu souboru klienta	ExitsDefaultPath
ClientExitSekce Cesta ke konfiguračnímu souboru klienta	ExitsDefaultPath64
ClientExitSekce Cesta ke konfiguračnímu souboru klienta	Cesta ke třídě JavaExits
stanza JMQI konfiguračního souboru klienta	useMQCSPauthentication
stanzaMessageBuffer konfiguračního souboru klienta	MaximumSize

Tabulka 54. Který oddíl konfiguračního souboru klienta obsahuje který atribut (pokračování)	
sekce	**Atribut**
stanzaMessageBuffer konfiguračního souboru klienta	PurgeTime
stanzaMessageBuffer konfiguračního souboru klienta	UpdatePercentage
Sekce TCP konfiguračního souboru klienta	ClntRcvBuffSize
Sekce TCP konfiguračního souboru klienta	ClntSndBuffSize
Sekce TCP konfiguračního souboru klienta	Časový limit připojení
Sekce TCP konfiguračního souboru klienta	KeepAlive

Další informace o konfiguraci produktu IBM MQ MQI client naleznete v tématu Konfigurace klienta pomocí konfiguračního souboru.

Související informace

Trasování tříd produktu IBM MQ pro aplikace v jazyce Java

stanza Java Standardní trasování prostředí

Pomocí oddílu Nastavení trasování standardního prostředí produktu Java lze konfigurovat prostředek trasování produktu IBM MQ classes for Java .

com.ibm.msg.client.commonservices.trace.outputName = *traceOutputNázev*

traceOutputName je adresář a název souboru, do kterého je odeslán výstup trasování.

Při výchozím nastavení jsou trasovací informace zapsány do trasovacího souboru v aktuálním pracovním adresáři aplikace. Název trasovacího souboru závisí na prostředí, v němž je aplikace spuštěna:

- Pro IBM MQ classes for Java for IBM MQ 9.0.0 Fix Pack 1 nebo starší, je trasování zapsáno do souboru s názvem `mqjms_%PID%.trc`.
- V 9.0.0.2 Pokud v produktu IBM MQ 9.0.0 Fix Pack 2byla aplikace načtena ze souboru JAR `com.ibm.mq.jar`, je zapsána do souboru s názvem `mqjava_%PID%.trc`, pokud aplikace má hodnotu IBM MQ classes for Java .
- V 9.0.0.2 Pokud v produktu IBM MQ 9.0.0 Fix Pack 2byla aplikace načtena IBM MQ classes for Java z přemístitelného souboru JAR `com.ibm.mq.allclient.jar`, je trasovací soubor zapsán do souboru s názvem `mqjavaclient_%PID%.trc`.
- V 9.0.0.10 Pokud v produktu IBM MQ 9.0.0 Fix Pack 10byla aplikace načtena ze souboru JAR `com.ibm.mq.jar`, je zapsána do souboru s názvem `mqjava_%PID%.cl%u.trc`, pokud aplikace má hodnotu IBM MQ classes for Java .
- V 9.0.0.10 Pokud v produktu IBM MQ 9.0.0 Fix Pack 10byla aplikace načtena IBM MQ classes for Java z přemístitelného souboru JAR `com.ibm.mq.allclient.jar`, je trasovací soubor zapsán do souboru s názvem `mqjavaclient_%PID%.cl%u.trc`.

kde *%PID%* je identifikátor procesu trasované aplikace, a *%u* je jedinečné číslo pro rozlišení souborů mezi podprocesy spuštěnými trasováním v různých zavaděčích tříd Java.

Uvedete-li alternativní adresář, musí existovat a musíte mít oprávnění k zápisu do tohoto adresáře. Nemáte-li oprávnění k zápisu, bude výstup trasování zapsán do `System.err`.

com.ibm.msg.client.commonservices.trace.include = *includeList*

includeList je seznam balíků a tříd, které jsou trasovány, nebo speciální hodnoty `ALL` nebo `NONE`.

Oddělte názvy balíků nebo tříd středníkem, `;`. *includeList* standardně zobrazuje `ALL`a trasuje všechny balíky a třídy v IBM MQ classes for Java.

Poznámka: Můžete zahrnout balík, ale pak vyloučit podbalíky tohoto balíku. Pokud například zahrnete balík `a.b` a vyloučíte balík `a.b.x`, trasování zahrnuje vše v `a.b.y` a `a.b.z`, ale ne `a.b.x` nebo `a.b.x.1`.

com.ibm.msg.client.commonservices.trace.exclude = *excludeList*
excludeList je seznam balíků a tříd, které nejsou trasovány, nebo speciální hodnoty `ALL` nebo `NONE`.

Oddělte názvy balíků nebo tříd středníkem, `;`. *excludeList* standardně zobrazuje `NONE`, a proto vylučuje žádné balíky a třídy v IBM MQ classes for JMS , které jsou trasovány.

Poznámka: Balík můžete vyloučit, ale pak zahrnout podbalíky tohoto balíku. Například, pokud vyloučíte balík `a.b` a zahrnete balík `a.b.x`, trasování zahrnuje vše v `a.b.x` a `a.b.x.1`, ale ne `a.b.y` nebo `a.b.z`.

Zahrne se jakýkoli balík nebo třída, která je uvedená, na stejné úrovni, jak je zahrnuta i vyloučená.

com.ibm.msg.client.commonservices.trace.maxBytes = *maxArrayBajty*
maxArrayBytes je maximální počet bajtů, které jsou trasovány z libovolných bajtových polí.

Je-li hodnota *maxArrayBytes* nastavena na kladné celé číslo, omezuje počet bajtů v bajtovém poli, které jsou zapsány do trasovacího souboru. Ořízne bajtové pole po zapsání *maxArrayBytes* ven. Nastavení *maxArrayBytes* snižuje velikost výsledného trasovacího souboru a snižuje účinek trasování na výkon aplikace.

Hodnota `0` této vlastnosti znamená, že žádný obsah žádného bajtového pole není odeslán do trasovacího souboru.

Předvolená hodnota je `-1`, která odstraňuje jakékoli omezení počtu bajtů v bajtovém poli, které se posílají do trasovacího souboru.

com.ibm.msg.client.commonservices.trace.limit = *maxTraceBajty*
maxTraceBytes je maximální počet bajtů, které jsou zapsány do výstupního trasovacího souboru.

Produkt *maxTraceBytes* pracuje s produktem *traceCycles*. Pokud se počet bajtů trasování nachází v blízkosti limitu, soubor se zavře a spustí se nový výstupní soubor trasování.

Hodnota `0` znamená, že výstupní trasovací soubor má nulovou délku. Předvolená hodnota je `-1`, což znamená, že množství dat, která mají být zapsána do výstupního souboru trasování, je neomezeno.

com.ibm.msg.client.commonservices.trace.count = *traceCycles*
traceCycles je počet výstupních souborů trasování, které se mají procházet.

Pokud aktuální trasovací výstupní soubor dosáhne limitu zadaného pomocí *maxTraceBytes*, soubor se zavře. Další výstup trasování se zapisuje do dalšího výstupního souboru trasování v pořadí. Každý trasovací výstupní soubor se odlišuje od číselné přípony připojené k názvu souboru. Aktuální nebo poslední trasovací výstupní soubor je `mqjms.trc.0`, další nejnovější trasovací výstupní soubor je `mqjms.trc.1`. Starší trasovací soubory postupují podle stejného vzoru číslování až do limitu.

Standardní hodnota *traceCycles* je 1. Je-li hodnota *traceCycles* 1, dosáhne-li aktuální výstupní soubor trasování jeho maximální velikost, bude soubor uzavřen a odstraněn. Je spuštěn nový výstupní soubor trasování se stejným názvem. Proto v daném okamžiku existuje pouze jeden výstupní soubor trasování.

com.ibm.msg.client.commonservices.trace.parameter = *traceParameters*
Produkt *traceParameters* řídí, zda jsou parametry metod a návratové hodnoty zahrnuty do trasování.

Výchozí hodnota parametru *traceParameters* je TRUE. Je-li parametr *traceParameters* nastaven na hodnotu FALSE, trasují se pouze podpisy metody.

com.ibm.msg.client.commonservices.trace.startup = *spuštění*
Existuje inicializační fáze IBM MQ classes for Java , během které jsou prostředky alokovány. Hlavní prostředek trasování je inicializován během fáze přidělování prostředků.

Je-li parametr *startup* nastaven na hodnotu TRUE, je použito trasování spouštění. Informace o trasování se vytvoří okamžitě a zahrnují nastavení všech komponent včetně samotného trasovacího

prostředku. Informace o trasování spuštění lze použít k diagnostice problémů s konfigurací. Informace o trasování při spuštění se vždy zapisují do `System.err`.

Výchozí hodnota parametru *`startup`* je FALSE.

`startup` je zkontrolováno před dokončením inicializace. Z tohoto důvodu specifikujte pouze vlastnost na příkazovém řádku jako systémovou vlastnost Java . Neuvádějte jej do konfiguračního souboru IBM MQ classes for Java .

com.ibm.msg.client.commonservices.trace.compress = *compressedTrace*
Chcete-li komprimovat trasovací výstup, nastavte *`compressedTrace`* na TRUE .

Standardní hodnota *`compressedTrace`* je FALSE.

Je-li parametr *`compressedTrace`* nastaven na hodnotu TRUE, je výstup trasování komprimován. Výchozí název výstupního souboru trasování má příponu `.trz`. Je-li komprese nastavena na FALSE, výchozí hodnota, má soubor příponu `.trc` , aby indikoval, že je nekomprimovaný. Pokud však název souboru pro výstup trasování byl zadán v souboru *`traceOutputName`* , bude místo něj použit název. Pro tento soubor se nepoužije žádná přípona.

Komprimovaný výstup trasování je menší než nekomprimovaný. Protože je zde méně vstupů/výstupů, lze jej zapsat rychleji než nekomprimované trasování. Komprimované trasování má menší vliv na výkon IBM MQ classes for Java než nekomprimované trasování.

Je-li nastavena hodnota *`maxTraceBytes`* a *`traceCycles`* , vytvoří se více komprimovaných trasovacích souborů místo několika prostých textových souborů.

Pokud IBM MQ classes for Java končí neřízeným způsobem, komprimovaný trasovací soubor nemusí být platný. Z tohoto důvodu musí být komprese trasování použita pouze v případě, že se IBM MQ classes for Java vypíná řízeným způsobem. Kompresi trasování používejte pouze v případě, že problémy, které jsou předmětem šetření, nezpůsobují neočekávané zastavení prostředí JVM. Nepoužívejte kompresi trasování při diagnostice problémů, které mohou způsobit `System.Halt()` ukončení činnosti nebo nestandardní, nekontrolované ukončení prostředí JVM.

com.ibm.msg.client.commonservices.trace.level = *traceLevel*
`traceLevel` uvádí úroveň filtrování pro trasování. Definovaná úroveň trasování je následující:

- `TRACE_NONE`: `0`
- `TRACE_EXCEPTION`: `1`
- `TRACE_WARNING`: `3`
- `TRACE_INFO`: `6`
- `TRACE_ENTRYEXIT`: `8`
- `TRACE_DATA`: `9`
- `TRACE_ALL`: `Integer.MAX_VALUE`

Každá úroveň trasování zahrnuje všechny nižší úrovně. Je-li například úroveň trasování nastavena na hodnotu `TRACE_INFO`, je do trasování zapisován libovolný trasovací bod s definovanou úrovní `TRACE_EXCEPTION`, `TRACE_WARNING`nebo `TRACE_INFO` . Všechny ostatní stopové body jsou vyloučeny.

com.ibm.msg.client.commonservices.trace.standalone = *standaloneTrace*

`standaloneTrace` řídí, zda je služba trasování klienta IBM MQ classes for Java použita v prostředí WebSphere Application Server .

Je-li parametr *`standaloneTrace`* nastaven na hodnotu TRUE, jsou pro určení konfigurace trasování použity vlastnosti trasování klienta produktu IBM MQ classes for Java .

Je-li parametr *`standaloneTrace`* nastaven na hodnotu FALSEa klient IBM MQ classes for Java je spuštěn v kontejneru WebSphere Application Server , použije se trasovací služba WebSphere Application Server . Trasovací informace, které se vygenerují, závisí na nastavení trasování aplikačního serveru.

Standardní hodnota *`standaloneTrace`* je FALSE.

IBM MQ classes for Java a nástroje pro správu softwaru
Nástroje pro správu softwaru, jako např. Apache Maven, lze použít s produktem IBM MQ classes for Java.

Řada velkých rozvojových organizací používá tyto nástroje k centrální správě úložišť knihoven jiných dodavatelů.

IBM MQ classes for Java se skládá z určitého počtu souborů JAR. Když vyvíjíte jazykové aplikace produktu Java pomocí tohoto rozhraní API, je na počítači, na kterém je vyvíjována aplikace, vyžadována instalace buď produktu IBM MQ Server, klienta IBM MQ , nebo klienta IBM MQ SupportPac .

Chcete-li použít nástroj pro správu softwaru a přidat soubory JAR, které tvoří IBM MQ classes for Java do centrálně spravovaného úložiště, musí být dodrženy následující body:

- Úložiště nebo kontejner musí být k dispozici pouze pro vývojáře v rámci vaší organizace. Jakékoli rozdělení mimo organizaci není povoleno.
- Úložiště musí obsahovat úplnou a konzistentní sadu souborů JAR z jedné verze produktu IBM MQ nebo z opravné sady Fix Pack.
- Jste zodpovědní za aktualizaci úložiště pomocí jakékoli údržby poskytované produktem IBM Support.

V produktu IBM MQ 8.0musí být soubor JAR produktu `com.ibm.mq.allclient.jar` instalován do úložiště.

V produktu IBM MQ 9.0jsou vyžadovány soubory JAR Bouncy Castle Security a soubory JAR podpory CMS. Další informace viz “Co je nainstalováno pro IBM MQ classes for Java” na stránce 313 a Podpora pro jiná prostředí nežIBM JRE.

Postup po instalaci pro aplikace produktu IBM MQ classes for Java
Po instalaci produktu IBM MQ classes for Javamůžete nakonfigurovat svou instalaci tak, abyste spouštěli vlastní aplikace.

Nezapomeňte zkontrolovat soubor Readme produktu IBM MQ pro nejnovější informace nebo více specifických informací o vašem prostředí. Nejnovější verze souboru Readme k produktu je k dispozici na webové stránce Přečtené položky produktu IBM MQ, WebSphere MQa MQSeries .

Před pokusem o spuštění aplikace IBM MQ classes for Java v režimu vazeb se ujistěte, že jste nakonfigurovali produkt IBM MQ podle popisu v části Konfigurace.

Konfigurace správce front tak, aby přijímal klientská připojení z produktu IBM MQ classes for Java
Chcete-li nakonfigurovat správce front tak, aby přijímal příchozí požadavky na připojení od klientů, definujte a povolte použití kanálu připojení k serveru a spusťte program modulu listener.

Podrobnosti viz “Konfigurace správce front pro příjem klientských připojení na platformách Multiplatforms” na stránce 1045.

Spuštění aplikací produktu IBM MQ classes for Java v rámci struktury Java Security Manager
Produkt IBM MQ classes for Java může být spuštěn s povoleným prostorem Java Security Manager . Chcete-li úspěšně spustit aplikace s povoleným prostorem Java Security Manager , musíte nakonfigurovat prostředí Java virtual machine (JVM) s vhodným souborem definic zásad.

Nejjednodušším způsobem, jak vytvořit vhodný definiční soubor zásad, je změna souboru zásad dodaného s Java runtime environment (JRE). Na většině systémů je tento soubor uložen v adresáři `path lib/security/java.policy`, relativně k adresáři prostředí JRE. Soubory zásad můžete upravit buď pomocí svého preferovaného editoru, nebo pomocí programu nástroje politačního programu dodaného s vaším prostředím JRE.

Musíte udělit oprávnění k souboru `com.ibm.mq.jmqi.jar` , aby mohl:

- Vytvoření soketů (v režimu klienta)
- Načíst nativní knihovnu (v režimu vazeb)
- Číst různé vlastnosti z prostředí

Vlastnost systému **`os.name`** musí být k dispozici pro IBM MQ classes for Java , pokud běží pod Java Security Manager.

Správce front produktu IBM MQ může odesílat oznámení připojeným klientům, kteří požadují řízené ukončení konverzací (manipulátory připojení), například když je správce front uveden do klidového stavu. Pokud podproces v rámci klienta produktu Java obdrží jedno z těchto oznámení současně s jiným podprocesem v rámci klienta, požádá o novou konverzaci, může dojít k uváznutí, protože obě podprocesy potřebují přístup k internímu "connectionsLock" na objektu specifikace RemoteConnection.

V 9.0.0.3 V 9.0.5 Od IBM MQ 9.0.0 Fix Pack 3 a IBM MQ 9.0.5je uváznutí klienta IBM MQ Java pevné. Pokud vaše aplikace Java používá Java Security Manager, musíte přidat následující oprávnění k souboru `java.security.policy` používanu aplikací, jinak dojde k výjimce v aplikaci:

```
permission java.lang.RuntimePermission "modifyThread";
```

Tento parametr RuntimePermission je klientem vyžadován jako součást správy přiřazení a uzavření multiplexních konverzací v rámci připojení prostřednictvím protokolu TCP/IP ke správcům front.

Příklad zadání záznamu souboru zásad

Zde je uveden příklad položky souboru zásad, která umožňuje, aby produkt IBM MQ classes for Java byl úspěšně spuštěn pod výchozím správcem zabezpečení. Řetězec *MQ_INSTALLATION_PATH* nahraďte v tomto příkladu umístěním, kde je ve vašem systému nainstalován produkt IBM MQ classes for Java .

```
grant codeBase "file: MQ_INSTALLATION_PATH/java/lib/*" {
//We need access to these properties, mainly for tracing
permission java.util.PropertyPermission "user.name","read";
permission java.util.PropertyPermission "os.name","read";
permission java.util.PropertyPermission "user.dir","read";
permission java.util.PropertyPermission "line.separator","read";
permission java.util.PropertyPermission "path.separator","read";
permission java.util.PropertyPermission "file.separator","read";
permission java.util.PropertyPermission "com.ibm.msg.client.commonservices.log.*","read";
permission java.util.PropertyPermission "com.ibm.msg.client.commonservices.trace.*","read";
permission java.util.PropertyPermission "Diagnostics.Java.Errors.Destination.Filename","read";
permission java.util.PropertyPermission "com.ibm.mq.commonservices","read";
permission java.util.PropertyPermission "com.ibm.mq.cfg.*","read";

//Tracing - we need the ability to control java.util.logging
permission java.util.logging.LoggingPermission "control";
// And access to create the trace file and read the log file - assumed to be in the current
directory
permission java.io.FilePermission "*","read,write";

// Required to allow a trace file to be written to the filesystem.
// Replace 'TRACE_FILE_DIRECTORY' with the directory name where trace is to be written to
permission java.io.FilePermission "TRACE_FILE_DIRECTORY","read,write";
permission java.io.FilePermission "TRACE_FILE_DIRECTORY/*","read,write";

// We'd like to set up an mBean to control trace
permission javax.management.MBeanServerPermission "createMBeanServer";
permission javax.management.MBeanPermission "*","*";

// We need to be able to read manifests etc from the jar files in the installation directory
permission java.io.FilePermission "MQ_INSTALLATION_PATH/java/lib/-","read";

//Required if mqclient.ini/mqs.ini configuration files are used
permission java.io.FilePermission "MQ_DATA_DIRECTORY/mqclient.ini","read";
permission java.io.FilePermission "MQ_DATA_DIRECTORY/mqs.ini","read";

//For the client transport type.
permission java.net.SocketPermission "*","connect,resolve";

//For the bindings transport type.
permission java.lang.RuntimePermission "loadLibrary.*";

//For applications that use CCDT tables (access to the CCDT AMQCLCHL.TAB)
permission java.io.FilePermission "MQ_DATA_DIRECTORY/qmgrs/QM_NAME/@ipcc/AMQCLCHL.TAB","read";

//For applications that use User Exits
permission java.io.FilePermission "MQ_DATA_DIRECTORY/exits/*","read";
permission java.io.FilePermission "MQ_DATA_DIRECTORY/exits64/*","read";
permission java.lang.RuntimePermission "createClassLoader";

//Required for the z/OS platform
permission java.util.PropertyPermission "com.ibm.vm.bitmode","read";
```

```
// Used by the internal ConnectionFactory implementation
permission java.lang.reflect.ReflectPermission "suppressAccessChecks";

// Used for controlled class loading
permission java.lang.RuntimePermission "setContextClassLoader";

// Used to default the Application name in Client mode connections
permission java.util.PropertyPermission "sun.java.command","read";

// Used by the IBM JSSE classes
permission java.util.PropertyPermission "com.ibm.crypto.provider.AESNITrace","read";

//Required to determine if an IBM Java Runtime is running in FIPS mode,
//and to modify the property values status as required.
permission java.util.PropertyPermission "com.ibm.jsse2.usefipsprovider","read,write";
permission java.util.PropertyPermission "com.ibm.jsse2.JSSEFIPS","read,write";
//Required if an IBM FIPS provider is to be used for SSL communication.
permission java.security.SecurityPermission "insertProvider.IBMJCEFIPS";

// Required for non-IBM Java Runtimes that establish secure client
// transport mode connections using mutual TLS authentication
permission java.util.PropertyPermission "javax.net.ssl.keyStore","read";
permission java.util.PropertyPermission "javax.net.ssl.keyStorePassword","read";

V 9.0.0.3// Required for Java applications that use the Java Security Manager
permission java.lang.RuntimePermission "modifyThread";
};
```

Tento příklad souboru zásad umožňuje produktu IBM MQ classes for Java pracovat správně pod správcem zabezpečení, ale přesto může být nutné, aby váš vlastní kód fungoval správně, než budete moci aplikace pracovat.

Vzorový kód dodaný s IBM MQ classes for Java nebyl specificky povolen pro použití se správcem zabezpečení; avšak testy IVT se spustí s tímto souborem zásad a výchozím správcem zabezpečení na místě.

Spuštění aplikací produktu IBM MQ classes for Java v rámci komponenty CICS Transaction Server
Aplikaci IBM MQ classes for Java lze spustit jako transakci pod serverem CICS Transaction Server.

Chcete-li spustit aplikaci IBM MQ classes for Java jako transakci pod serverem CICS Transaction Server pro produkt z/OS, proveďte následující kroky:

1. Definujte aplikaci a transakci pomocí produktu CICS pomocí zadané transakce CEDA.
2. Ujistěte se, že je adaptér IBM MQ CICS nainstalován ve vašem systému CICS . z/OS (Podrobnosti najdete v tématu Použití IBM MQ s CICS .)
3. Ujistěte se, že prostředí JVM uvedené v produktu CICS obsahuje odpovídající položky CLASSPATH a LIBPATH.
4. Zahajte transakci pomocí libovolného z vašich běžných procesů.

Další informace o spouštění transakcí produktu CICS Java naleznete v dokumentaci k systému CICS .

Ověření instalace produktu IBM MQ classes for Java

Program pro ověření instalace, produkt MQIVP, je dodáván s produktem IBM MQ classes for Java. Tento program můžete použít k testování všech režimů připojení produktu IBM MQ classes for Java.

Program vás vyzve k zadání řady voleb a jiných dat, abyste určili, který režim připojení chcete ověřit. K ověření instalace použijte následující proceduru:

1. Chcete-li spustit program v režimu klienta, nakonfigurujte správce front tak, jak je popsáno v tématu “Konfigurace správce front pro příjem klientských připojení na platformách Multiplatforms” na stránce 1045. Fronta, která má být použita, je SYSTEM.DEFAULT.LOCAL.QUEUE.
2. Chcete-li spustit program v režimu klienta, přečtěte si také téma “použitíIBM MQ classes for Java” na stránce 308.

   Proveďte zbývající kroky tohoto postupu na systému, na kterém chcete spustit program.

3. Ujistěte se, že jste aktualizovali proměnnou prostředí CLASSPATH podle pokynů v části "Proměnné prostředí relevantní pro produkt IBM MQ classes for Java" na stránce 316.
4. Změňte adresář na *MQ_INSTALLATION_PATH*/mqm/samp/wmqjava/samples, kde *MQ_INSTALLATION_PATH* je cesta k vaší instalaci produktu IBM MQ . Pak na příkazový řádek zadejte:

   ```
   java -Djava.library.path= library_path MQIVP
   ```

   kde *cesta_knihovny* je cesta ke knihovnám produktu IBM MQ classes for Java (viz "IBM MQ classes for Java knihovny" na stránce 318).

   Na výzvu označenou (1):

   - Chcete-li použít připojení TCP/IP, zadejte název hostitele serveru IBM MQ .
   - Chcete-li použít nativní připojení (režim vazeb), ponechte pole prázdné (nezadávejte název).

   Program se pokouší:

   - 1. Připojte se ke správci front
   - 2. Otevřete frontu SYSTEM.DEFAULT.LOCAL.QUEUE, vložte zprávu do fronty, získat zprávu z fronty a poté zavřít frontu
   - 3. Odpojení od správce front
   - 4. Vrátit zprávu, pokud jsou operace úspěšné

Zde je uveden příklad výzev a odpovědí, které můžete vidět. Skutečné výzvy k zadání a vaše odpovědi závisí na vaší síti IBM MQ .

```
Please enter the IP address of the MQ server    : ipaddress(1)
Please enter the port to connect to             : (1414) (2)
Please enter the server connection channel name : channelname (2)
Please enter the queue manager name             : qmname
Success: Connected to queue manager.
Success: Opened SYSTEM.DEFAULT.LOCAL.QUEUE
Success: Put a message to SYSTEM.DEFAULT.LOCAL.QUEUE
Success: Got a message from SYSTEM.DEFAULT.LOCAL.QUEUE
Success: Closed SYSTEM.DEFAULT.LOCAL.QUEUE
Success: Disconnected from queue manager


Tests complete -
SUCCESS: This MQ Transport is functioning correctly.
Press Enter to continue ...
```

Poznámka:

1. z/OS V systému z/OSponechte pole prázdné na výzvu k zadání (1).
2. Vyberete-li připojení k serveru, nezobrazí se výzvy k zadání (2).
3. IBM i V systému IBM imůžete zadat příkaz `java MQIVP` pouze z prostředí QShell. Případně můžete aplikaci spustit pomocí příkazu CL `RUNJVA CLASS(MQIVP)`.

Použití ukázkových aplikací produktu IBM MQ classes for Java

Ukázkové aplikace produktu IBM MQ classes for Java poskytují přehled běžných funkcí rozhraní API produktu IBM MQ classes for Java . Můžete je použít k ověření instalace a serveru systému zpráv, které vám pomohou při sestavování svých vlastních aplikací.

Informace o této úloze

Potřebujete-li pomoci při vytváření vlastních aplikací, můžete ukázkové aplikace použít jako výchozí bod. Pro každou aplikaci je k dispozici jak zdrojový, tak i kompilovaná verze. Zkontrolujte ukázkový zdrojový kód a určete klíčové kroky k vytvoření jednotlivých vyžadovaných objektů pro danou aplikaci (MQQueueManager, MQConstants, MQMessage, MQPutMessagea MQDestination) a pro nastavení specifických vlastností, které jsou zapotřebí k určení způsobu, jakým má aplikace pracovat. Další

informace viz “Zápis aplikací IBM MQ classes for Java” na stránce 332. Ukázky mohou být předmětem změn v budoucích vydáních produktu IBM MQ Java.

Produkt Tabulka 55 na stránce 329 zobrazuje, kde jsou ukázkové aplikace produktu IBM MQ classes for Java instalovány na jednotlivých platformách:

Tabulka 55. Instalační adresáře pro ukázkové aplikace produktu IBM MQ classes for Java

Platforma	Adresář
UNIX UNIX Linux Linux	*MQ_INSTALLATION_PATH*/samp/wmqjava/samples
Windows Windows	*MQ_INSTALLATION_PATH*\tools\wmqjava\samples
IBM i IBM i	/qibm/proddata/mqm/java/samples/wmqjava/samples
z/OS z/OS	*MQ_INSTALLATION_PATH*/java/samples/wmqjava

Tabulka 56 na stránce 329 zobrazuje sady ukázkových aplikací, které jsou dodávány s produktem IBM MQ classes for Java.

Tabulka 56. IBM MQ classes for Java ukázkové aplikace

Název vzorku	Popis
IMSBridgeSample.java	Jednoduchý program demonstrující použití mostu IMS se serverem IBM MQ classes for Java.
MQIVP.java	Ověřovací program instalace produktu IBM MQ Java .
MQMessagePropertiesSample.java	Předvádí použití rozhraní API Message Properties zavedeného do IBM WebSphere MQ 7.0.
MQPubSubApiSample.java	Demonstruje použití rozhraní API publish/odběru zavedeného do produktu IBM WebSphere MQ 7.0.
MQSample.java	Jednoduchý program k demonstraci vložení a získání zprávy z fronty.
MQSampleMessageManager.java	Utility class for message handling in the IBM MQ base Java samples.
mqjcivp.properties	Tento balík prostředků obsahuje zprávy, které používá program pro ověřování instalace produktu IBM MQ classes for Java (`MQIVP.java`).

Produkt IBM MQ classes for Java nabízí skript s názvem `runjms` , který lze použít ke spouštění ukázkových aplikací. Tento skript nastaví prostředí produktu IBM MQ , aby bylo možné spustit ukázkové aplikace produktu IBM MQ classes for Java .

Tabulka 57 na stránce 329 zobrazuje umístění skriptu na každé platformě:

Tabulka 57. Umístění skriptu `runjms`

Platforma	Adresář
UNIX UNIX Linux Linux	*MQ_INSTALLATION_PATH*/java/bin/runjms

Tabulka 57. Umístění skriptu runjms (pokračování)	
Platforma	**Adresář**
Windows	*MQ_INSTALLATION_PATH*\java\bin\runjms.bat
IBM i	/qibm/proddata/mqm/java/bin/runjms , nebo /qibm/proddata/mqm/java/bin/runjms64
z/OS	*MQ_INSTALLATION_PATH*java/bin/runjms

Chcete-li použít skript runjms k vyvolání ukázkové aplikace, proveďte následující kroky:

Postup

1. Vyveďte příkazový řádek a přejděte do adresáře obsahujícího ukázkovou aplikaci, kterou chcete spustit.
2. Zadejte následující příkaz:

   ```
   Path to the runjms script/runjms sample_application_name
   ```

   Ukázková aplikace zobrazí seznam parametrů, které potřebuje.
3. Chcete-li spustit ukázku s těmito parametry, zadejte následující příkaz:

   ```
   Path to the runjms script/runjms sample_application_name parameters
   ```

Příklad

Linux Chcete-li například spustit ukázku MQIVP v systému Linux, zadejte následující příkazy:

```
cd /opt/mqm/samp/wmqjava/samples
/opt/mqm/java/bin/runjms MQIVP
```

Související pojmy

“Co je nainstalováno pro IBM MQ classes for JMS” na stránce 74
Při instalaci produktu IBM MQ classes for JMSse vytvoří mnoho souborů a adresářů. V systému Windowsse při instalaci provádí některá konfigurace automaticky nastavováním proměnných prostředí. Na jiných platformách a v některých prostředích produktu Windows musíte nastavit proměnné prostředí před spuštěním aplikací produktu IBM MQ classes for JMS .

Řešení problémů s produktem IBM MQ classes for Java

Na začátku spusťte program pro ověření instalace. Je možné, že budete muset použít trasovací prostředek.

Pokud se program nedokončí úspěšně, spusťte program pro ověření instalace a postupujte podle pokynů uvedených v diagnostických zprávách. Tento program je popsán v publikaci “Ověření instalace produktu IBM MQ classes for Java” na stránce 327.

Pokud problémy pokračují a potřebujete se obrátit na tým služeb IBM , můžete být požádáni o zapnutí trasovacího prostředku. Proveďte to tak, jak je uvedeno v následujícím příkladu.

Chcete-li trasovat program MQIVP :

- Vytvořte soubor vlastností *com.ibm.mq.commonservices* (viz Použití com.ibm.mq.commonservices.

- Zadejte následující příkaz:

```
java -Dcom.ibm.mq.commonservices=commonservices_properties_file java
-Djava.library.path= library_path MQIVP -trace
```

kde:

- *commonservices_properties_file* je cesta (včetně názvu souboru) k souboru vlastností *com.ibm.mq.commonservices* .
- *cesta_knihovny* je cesta ke knihovnám IBM MQ classes for Java (viz"IBM MQ classes for Java knihovny" na stránce 318).

Další informace o tom, jak používat trasování, najdete v tématu Trasování aplikací produktu IBM MQ classes for Java.

V 9.0.4 z/OS MQ Adv. VUE Připojitelnost klientů Java a JMS k dávkovým aplikacím spuštěným v systému z/OS

JMSnebo IBM MQ classes for Javase aplikace na systému z/OS může připojit ke správci front v systému z/OS, který má atribut **ADVCAP** (ENABLED) , a to pomocí připojení klienta.

Hodnota **ADVCAP** (ENABLED) se vztahuje pouze na správce front z/OS , který je licencován jako IBM MQ Advanced for z/OS, Value Unit Edition (viz IBM MQ product identifiers and export information), běží s **QMGRPROD** nastaveným na ADVANCEDVUE.

Další informace o příkazu **ADVCAP** a START QMGR naleznete v části DISPLAY QMGR , kde získáte další informace o produktu **QMGRPROD**.

Všimněte si, že dávka je jediným podporovaným prostředím; neexistuje žádná podpora pro JMS pro CICS nebo JMS pro IMS.

Aplikace IBM MQ classes for JMSnebo IBM MQ classes for Javase v produktu z/OS nemůže připojit pomocí připojení v režimu klienta ke správci front, který není spuštěn v produktu z/OS, ani ke správci front, který nemá volbu **ADVCAP** (ENABLED) .

Pokusí-li se aplikace JMS o připojení pomocí režimu klienta a aplikace není povolena, vydá se zpráva výjimky JMSFMQ0005 .

Pokud se aplikace IBM MQ classes for Java na z/OS pokusí o připojení pomocí režimu klienta a není to povoleno, vrátí se MQRC_ENVIRONMENT_ERROR .

Podpora produktu Advanced Message Security (AMS)

V 9.0.5

Z klientských aplikací IBM MQ 9.0.5, IBM MQ classes for JMS nebo IBM MQ classes for Java lze použít AMS při připojování ke správcům front IBM MQ Advanced for z/OS, Value Unit Edition na vzdálených systémech z/OS .

Nový typ úložiště klíčů, `jceracfks`, je podporován pouze v produktu `keystore.conf` v systému z/OS , kde:

- Předpona názvu vlastnosti je `jceracfks` a tato předpona názvu nerozlišuje velká a malá písmena.
- Úložiště klíčů je svazek klíčů RACF.
- Hesla se nepožadují a budou ignorována. Důvodem je to, že soubory keyrings RACF nepoužívají hesla.
- Určíte-li poskytovatele, poskytovatel musí být `IBMJCE`.

Když používáte produkt `jceracfks` s produktem AMS, musí být úložiště klíčů ve tvaru: `safkeyring://user/keyring`, kde:

- `safkeyring` je literál a tento název nerozlišuje velká a malá písmena.
- *user* je ID uživatele RACF, který vlastní svazek klíčů

- *keyring* je název svazku klíčů RACF a název svazku klíčů je citlivý na velikost písmen

V následujícím příkladu je použit standardní svazek klíčů AMS pro uživatele JOHNDOE:

```
jceracfks.keystore=safkeyring://JOHNDOE/drq.ams.keyring
```

Zápis aplikací IBM MQ classes for Java

Tato kolekce témat obsahuje informace, které vám pomohou při zápisu aplikací produktu Java pro interakci se systémy IBM MQ .

Chcete-li použít produkt IBM MQ classes for Java pro přístup k frontám produktu IBM MQ , zapiš Java aplikace, které obsahují volání, která vložila zprávy do front IBM MQ , a získaní z nich zprávy. Podrobnosti o jednotlivých třídách naleznete v tématu IBM MQ classes for Java .

Poznámka: Automatické opětovné připojení klienta není podporováno produktem IBM MQ classes for Java.

Rozhraní IBM MQ classes for Java

Procedurální programovací rozhraní produktu IBM MQ používá příkazy, které pracují s objekty. Programovací rozhraní produktu Java používá objekty, se kterými se můžete chovat voláním metod.

Procedurální programovací rozhraní produktu IBM MQ je založeno na příkazových slovech, jako jsou tyto:

```
MQBACK, MQBEGIN, MQCLOSE, MQCONN, MQDISC,
MQGET, MQINQ, MQOPEN, MQPUT, MQSET, MQSUB
```

Tato příkazová slova slouží jako argument pro obsluhu objektu IBM MQ , na kterém mají pracovat. Váš program se skládá ze sady objektů IBM MQ , které se při volání metod na těchto objektech chovají.

Když použijete procedurální rozhraní, odpojíte se od správce front pomocí volání MQDISC (Hconn, CompCode, Reason), kde *Hconn* je manipulátor pro správce front.

V rozhraní produktu Java je správce front reprezentován objektem třídy MQQueueManager. Odpojení od správce front zavoláte voláním metody disconnect () na dané třídě.

```
// declare an object of type queue manager
MQQueueManager queueManager=new MQQueueManager();
...
// do something...
...
// disconnect from the queue manager
queueManager.disconnect();
```

Režimy připojení produktu IBM MQ classes for Java

Způsob, jakým program IBM MQ classes for Java obsahuje určité závislosti na režimech připojení, které chcete použít.

Pokud používáte připojení klienta, existuje řada rozdílů oproti IBM MQ MQI client , ale je koncepčně podobná. Používáte-li režim vazeb, můžete použít vazby se zkrácenou cestou a můžete zadat příkaz MQBEGIN. Režim, který má být použit, určujete nastavením proměnných ve třídě MQEnvironment.

IBM MQ classes for Java připojení klientů

Když je IBM MQ classes for Java použit jako klient, je jako IBM MQ MQI client, ale má řadu rozdílů.

Pokud programujete *IBM MQ classes for Java* jako klienta pro použití jako klient, uvědomte si následující rozdíly:

- Podporuje pouze TCP/IP.
- Při spuštění se nečte žádné proměnné prostředí IBM MQ .

- Informace, které budou uloženy v definici kanálu a v proměnných prostředí, mohou být uloženy ve třídě s názvem Prostředí. Alternativně lze tyto informace předat jako parametry při vytvoření připojení.
- Chybové stavy a výjimky jsou zapsány do protokolu uvedeného ve třídě `MQException` . Výchozí místo určení chyb je konzola produktu Java .
- Pouze následující atributy v konfiguračním souboru klienta IBM MQ jsou relevantní pro produkt IBM MQ classes for Java. Určíte-li jiné atributy, jsou neúčinné.

sekce	**Atribut**
ClientExitSekce Cesta ke konfiguračnímu souboru klienta	ExitsDefaultPath
ClientExitSekce Cesta ke konfiguračnímu souboru klienta	ExitsDefaultPath64
ClientExitSekce Cesta ke konfiguračnímu souboru klienta	JavaExitsClasspath
stanzaMessageBuffer konfiguračního souboru klienta	MaximumSize
stanzaMessageBuffer konfiguračního souboru klienta	PurgeTime
stanzaMessageBuffer konfiguračního souboru klienta	UpdatePercentage
Sekce TCP konfiguračního souboru klienta	ClntRcvBuffSize
Sekce TCP konfiguračního souboru klienta	ClntSndBuffSize
Sekce TCP konfiguračního souboru klienta	Časový limit připojení
Sekce TCP konfiguračního souboru klienta	KeepAlive

- Při připojení ke správci front, který vyžaduje převod znakových dat, je nyní klient V7 Java schopen provést převod, pokud to správce front není schopen provést. Prostředí JVM klienta musí podporovat převod mezi CCSID klienta a ID správce front.
- Prostředí IBM MQ classes for Java nepodporuje automatické opětovné připojování klientů.

Je-li použit v režimu klienta, produkt *IBM MQ classes for Java* nepodporuje volání MQBEGIN.

Režim vazeb IBM MQ classes for Java
Vázací režim produktu IBM MQ classes for Java se liší od režimu klienta třemi hlavními způsoby.

Při použití v režimu vázání produkt IBM MQ classes for Java používá rozhraní JNI (Java Native Interface) k volání přímo do existujícího rozhraní API správce front, a nikoli prostřednictvím komunikace prostřednictvím sítě.

Aplikace, které používají produkt IBM MQ classes for Java v režimu vazeb, se standardně připojují ke správci front pomocí volby *ConnectOption*, MQCNO_STANDARD_BINDINGS.

Produkt IBM MQ classes for Java podporuje následující *ConnectOptions*:

- VAZBA MQCNO_FASTPATH_BINDING
- VAZBA MQCNO_STANDARD_BINDING
- CQCNO_SHARED_BINDING
- VAZBA MQCNO_ISOLATED_BINDING

Další informace o *ConnectOptions*viz “Připojení ke správci front pomocí volání MQCONNX” na stránce 707.

Režim vazeb podporuje volání MQBEGIN za účelem zahájení globálních pracovních jednotek koordinovaných správcem front na všech platformách kromě produktů IBM MQ for IBM i a IBM MQ for z/OS.

Většina parametrů poskytnutých třídou `MQEnvironment` není důležitá pro režim vazeb a je ignorována.

Definování připojení produktu IBM MQ classes for Java k použití
Typ připojení, který má být použit, je určen nastavením proměnných ve třídě MQEnvironment.

Jsou použity dvě proměnné:

MQEnvironment.properties
Typ připojení je určen hodnotou přidruženou k názvu klíče `CMQC.TRANSPORT_PROPERTY`. Možné hodnoty jsou následující:

CMQC.TRANSPORT_MQSERIES_BINDINGS
Připojit se v režimu vazeb

CMQC.TRANSPORT_MQSERIES_CLIENT
Připojit v režimu klienta

CMQC.TRANSPORT_MQSERIES
Režim připojení je určen hodnotou vlastnosti *hostname* .

MQEnvironment.hostname
Nastavte hodnotu této proměnné následujícím způsobem:

- U klientských připojení nastavte hodnotu této proměnné na název hostitele serveru IBM MQ , ke kterému se chcete připojit.
- Pro režim vazeb nenastavujte tuto proměnnou, nebo ji nastavte na hodnotu null.

Operace pro správce front
Tato kolekce témat popisuje, jak se připojit k správci front a odpojit od ní pomocí produktu IBM MQ classes for Java.

Nastavení prostředí produktu IBM MQ pro produkt IBM MQ classes for Java
Aby se aplikace mohla připojit ke správci front v režimu klienta, musí aplikace určit název kanálu, název hostitele a číslo portu.

Poznámka: Informace v tomto tématu jsou relevantní pouze v případě, že se vaše aplikace připojuje ke správci front v režimu klienta. Pokud se připojuje v režimu vazeb, není relevantní. Viz “Režimy připojení pro produkt IBM MQ classes for JMS” na stránce 92.

Můžete zadat název kanálu, název hostitele a číslo portu jedním ze dvou způsobů: buď jako pole ve třídě MQEnvironment, nebo jako vlastnosti objektu MQQueueManager .

Nastavíte-li pole ve třídě MQEnvironment, použijí se na celou aplikaci, kromě případů, kdy jsou přepsány transformační tabulkou vlastností. Chcete-li určit název kanálu a název hostitele v prostředí MQEnvironment, použijte následující kód:

```
MQEnvironment.hostname = "host.domain.com";
MQEnvironment.channel = "java.client.channel";
```

To je ekvivalentní nastavení proměnné prostředí **MQSERVER** :

```
"java.client.channel/TCP/host.domain.com".
```

Při výchozím nastavení se klienti produktu Java pokusí připojit k modulu listener produktu IBM MQ na portu 1414. Chcete-li určit jiný port, použijte následující kód:

```
MQEnvironment.port = nnnn;
```

kde nnnn je požadované číslo portu.

Předáte-li vlastnosti objektu správce front při jeho vytvoření, použijí se pouze pro daného správce front. Vytvořte položky v objektu Hashtable s klíči **`hostname`**, **`channel`**a volitelně s hodnotami

porta s příslušnými hodnotami. Chcete-li použít výchozí port, 1414, můžete vynechat položku **port** . Vytvořte objekt MQQueueManager pomocí konstruktoru, který přijímá hašovací tabulku vlastností.

Identifikace připojení ke správci front nastavením názvu aplikace

Aplikace může nastavit název, který identifikuje její připojení ke správci front. Tento název aplikace se zobrazí příkazem **DISPLAY CONN MQSC/PCF** (kde pole se nazývá **APPLTAG**). nebo v zobrazení Průzkumníka IBM MQ **Připojení aplikace** (kde se pole nazývá **App name**).

Délka názvů aplikací je omezena na 28 znaků, takže delší názvy jsou zkráceny. Není-li zadán název aplikace, je poskytnuta výchozí hodnota. Výchozí název je založen na vyvolání (hlavní) třídě, ale pokud tyto informace nejsou k dispozici, použije se text `WebSphere MQ Client for Java` .

Je-li použit název vyvolávající třídy, je v případě potřeby upraven tak, aby se odstranily úvodní názvy balíků. Je-li například třída vyvolání `com.example.MainApp`, použije se úplný název, ale pokud je třída vyvolání `com.example.dictionaryAndThesaurus.multilingual.mainApp`, použije se název `multilingual.mainApp` , protože se jedná o nejdelší kombinaci názvu třídy a názvu balíku nejvíce vpravo, který se vejde do dostupné délky.

Je-li název třídy delší než 28 znaků, je oříznut na vhodný. Například `com.example.mainApplicationForSecondTestCase` se stane `mainApplicationForSecondTest`.

Chcete-li nastavit název aplikace ve třídě MQEnvironment, přidejte název do hašovací tabulky MQEnvironment.properties s klíčem **MQConstants.APPNAME_PROPERTY**pomocí následujícího kódu:

```
MQEnvironment.properties.put(MQConstants.APPNAME_PROPERTY, "my_application_name");
```

Chcete-li nastavit název aplikace v hašovací tabulce vlastností, která se předává konstruktoru MQQueueManager , přidejte název do hašovací tabulky vlastností s klíčem **MQConstants.APPNAME_PROPERTY**.

Potlačení vlastností zadaných v konfiguračním souboru klienta IBM MQ

Konfigurační soubor klienta IBM MQ může také určovat vlastnosti, které se používají ke konfiguraci produktu IBM MQ classes for Java. Vlastnosti zadané v konfiguračním souboru IBM MQ MQI client se však používají pouze v případě, že se aplikace připojuje ke správci front v režimu klienta.

V případě potřeby můžete přepsat libovolný atribut v konfiguračním souboru IBM MQ některým z následujících způsobů. Volby jsou zobrazeny v pořadí podle priority.

- Nastavte systémovou vlastnost Java pro vlastnost konfigurace.
- Nastavte tuto vlastnost v mapě MQEnvironment.properties .
- V systému Java5 a novějších verzích nastavte systémovou proměnnou prostředí.

Pouze následující atributy v konfiguračním souboru klienta IBM MQ jsou relevantní pro produkt IBM MQ classes for Java. Pokud uvedete nebo přepisují jiné atributy, nemá žádný efekt.

sekce	Atribut
ClientExitSekce Cesta ke konfiguračnímu souboru klienta	ExitsDefaultPath
ClientExitSekce Cesta ke konfiguračnímu souboru klienta	ExitsDefaultPath64
ClientExitSekce Cesta ke konfiguračnímu souboru klienta	JavaExitsClasspath
stanzaMessageBuffer konfiguračního souboru klienta	MaximumSize

sekce	Atribut
stanzaMessageBuffer konfiguračního souboru klienta	PurgeTime
stanzaMessageBuffer konfiguračního souboru klienta	UpdatePercentage
Sekce TCP konfiguračního souboru klienta	ClntRcvBufSize
Sekce TCP konfiguračního souboru klienta	ClntSndBufSize
Sekce TCP konfiguračního souboru klienta	Časový limit připojení
Sekce TCP konfiguračního souboru klienta	KeepAlive

Připojení ke správci front v produktu IBM MQ classes for Java
Připojte se ke správci front vytvořením nové instance třídy MQQueueManager . Odpojení od správce front voláním metody disconnect ().

Nyní jste připraveni připojit se ke správci front vytvořením nové instance třídy MQQueueManager :

```
MQQueueManager queueManager = new MQQueueManager("qMgrName");
```

Chcete-li se odpojit od správce front, volejte metodu disconnect () ve správci front:

```
queueManager.disconnect();
```

Pokud zavoláte metodu odpojení, všechny otevřené fronty a procesy, ke kterým jste v daném správci front přistoupil, jsou zavřeny. Je však dobrým programovacím postupem, abyste tyto prostředky při jejich používání explicitně uzavřeli. Chcete-li to provést, použijte metodu close () na příslušných objektech.

Metody commit () a backout () ve správci front jsou ekvivalentní volání MQCMIT a MQBACK, které se používají s procedurálním rozhraním.

Použití tabulky definic kanálů klienta s IBM MQ classes for Java
Klientská aplikace produktu IBM MQ classes for Java může používat definice kanálů připojení klienta uložené v tabulce definic kanálů klienta (CCDT).

Jako alternativu k vytvoření definice kanálu připojení klienta nastavením určitých polí a vlastností prostředí ve třídě `MQEnvironment` nebo jejich předáním do `MQQueueManager` v hašovací tabulce vlastností může klientská aplikace IBM MQ classes for Java používat definice kanálů připojení klienta, které jsou uloženy v tabulce definic kanálů klienta. Tyto definice jsou vytvořeny příkazy skriptu IBM MQ Script (MQSC) nebo příkazy PCF (IBM MQ Programmable Command Format) nebo pomocí nástroje IBM MQ Explorer .

Když aplikace vytvoří objekt `MQQueueManager` , klient IBM MQ classes for Java prohledá tabulku definic kanálů klienta pro vhodnou definici kanálu připojení klienta a použije definici kanálu ke spuštění kanálu MQI. Další informace o tabulkách definic kanálů klienta a o tom, jak je vytvořit, najdete v tématu Tabulka definic kanálů klienta.

Chcete-li použít tabulku definic kanálů klienta, musí aplikace nejprve vytvořit objekt URL. Objekt URL zapouzdřuje adresu URL, která identifikuje název a umístění souboru obsahujícího tabulku definic kanálů klienta a určuje, jak lze k souboru přistupovat.

Pokud například soubor `ccdt1.tab` obsahuje tabulku definic kanálů klienta a je uložen ve stejném systému, v němž je aplikace spuštěna, může aplikace vytvořit objekt adresy URL následujícím způsobem:

```
java.net.URL chanTab1 = new URL("file:///home/admdata/ccdt1.tab");
```

Jako další příklad předpokládejme, že soubor `ccdt2.tab` obsahuje tabulku definic kanálů klienta a je uložen v systému, který se liší od tabulky, na které je aplikace spuštěna. Pokud k souboru lze přistupovat pomocí protokolu FTP, může aplikace vytvořit objekt adresy URL následujícím způsobem:

```
java.net.URL chanTab2 = new URL("ftp://ftp.server/admdata/ccdt2.tab");
```

Poté, co aplikace vytvoří objekt URL, může aplikace vytvořit objekt `MQQueueManager` pomocí jednoho z konstruktorů, který vezme objekt URL jako parametr. Příklad:

```
MQQueueManager mars = new MQQueueManager("MARS", chanTab2);
```

Tento příkaz způsobí, že klient produktu IBM MQ classes for Java přistupuje k tabulce definic kanálů klienta identifikovanou objektem adresy URL chanTab2, prohledá tabulku pro vhodnou definici kanálu připojení klienta a poté použije definici kanálu ke spuštění kanálu MQI pro správce front s názvem MARS.

Všimněte si následujících bodů, které se použijí, pokud aplikace používá tabulku definic kanálů klienta:

- Když aplikace vytvoří objekt `MQQueueManager` pomocí konstruktoru, který přijímá objekt URL jako parametr, nesmí být ve třídě `MQEnvironment` nastaven žádný název kanálu, a to buď jako pole, nebo jako vlastnost prostředí. Je-li nastaven název kanálu, klient IBM MQ classes for Java vydá `MQException`. Vlastnost pole nebo prostředí určující název kanálu je považován za nastavený, pokud jeho hodnota obsahuje jinou hodnotu než null, prázdný řetězec nebo řetězec obsahující všechny prázdné znaky.
- Parametr **queueManagerName** v konstruktoru `MQQueueManager` může mít jednu z následujících hodnot:
  - Název správce front
  - Hvězdička (*) následovaná názvem skupiny správců front
  - Hvězdička (*)
  - Null, prázdný řetězec, nebo řetězec obsahující všechny prázdné znaky

  Jedná se o tytéž hodnoty, které lze použít pro parametr **QMgrName** v rámci volání MQCONN vydaného aplikací klienta, která používá rozhraní MQI (Message Queue Interface). Další informace o významu těchto hodnot najdete v tématu“Přehled rozhraní fronty zpráv” na stránce 690.

  Pokud vaše aplikace používá fondy připojení, prohlédněte si téma“Řízení výchozího fondu připojení v produktu IBM MQ classes for Java” na stránce 356.
- Když klient IBM MQ classes for Java najde vhodnou definici kanálu pro připojení klienta v tabulce definic kanálů klienta, použije pouze informace extrahované z této definice kanálu ke spuštění kanálu MQI. Všechny pole související s kanálem nebo vlastnosti prostředí, které aplikace může mít nastaveny ve třídě `MQEnvironment` , se budou ignorovat.

  Zejména si všimněte následujících bodů, pokud používáte Transport Layer Security (TLS):
  - Kanál MQI používá zabezpečení TLS pouze v případě, že definice kanálu extrahovaná z definiční tabulky kanálu klienta určuje název CipherSpec podporovaný klientem IBM MQ classes for Java .
  - Tabulka definic kanálů klienta také obsahuje informace o umístění serverů LDAP (Lightweight Directory Access Protocol), které uchovávají seznamy zrušených certifikátů (CRL). Klient IBM MQ classes for Java používá pouze tyto informace pro přístup k serverům LDAP, které obsahují CRL.
  - Definiční tabulka kanálu klienta může také obsahovat umístění odpovídacího modulu OCSP. Produkt IBM MQ classes for Java nemůže používat informace OCSP v souboru s tabulkou definic kanálů klienta. Nicméně můžete OCSP nakonfigurovat podle popisu v části Použití protokolu Online Certificate Protocol .

  Další informace o použití zabezpečení TLS s tabulkou definic kanálů klienta naleznete v tématu Určení, zda kanál MQI používá TLS.

  Všimněte si také následujících bodů, pokud používáte uživatelské procedury kanálu:

- Kanál MQI používá uživatelské procedury kanálu a přidružená uživatelská data určená prostřednictvím definice kanálu extrahované z tabulky definic kanálů klienta v preferovaném kanálu a data zadaná pomocí jiných metod.
- Definice kanálu extrahovaná z tabulky definic kanálů klienta může určovat uživatelské procedury kanálu, které jsou zapsány v systému Java, C nebo C + +. Další informace o tom, jak zapisovat uživatelskou proceduru kanálu v produktu Java , naleznete v tématu"Vytvoření uživatelské procedury kanálu v produktu IBM MQ classes for Java" na stránce 350. Další informace o tom, jak zapisovat uživatelskou proceduru kanálu v jiných jazycích, najdete v tématu "Použití uživatelských procedur kanálu nezapsaných v Java s IBM MQ classes for Java" na stránce 353.

Určení rozsahu portů pro připojení klienta IBM MQ classes for Java
Můžete uvést port, nebo rozsah portů, které může aplikace svázat dvěma způsoby.

Pokusí-li se aplikace IBM MQ classes for Java o připojení ke správci front produktu IBM MQ v režimu klienta, brána firewall může povolit pouze ta připojení, která pocházejí ze zadaných portů nebo rozsahu portů. V této situaci můžete uvést port nebo rozsah portů, ke kterým se aplikace může vázat. Port (y) můžete zadat následujícími způsoby:

- Můžete nastavit pole localAddressSetting ve třídě MQEnvironment. Příklad:

```
MQEnvironment.localAddressSetting = "192.0.2.0(2000,3000)";
```

- Můžete nastavit vlastnost prostředí CMQC.LOCAL_ADDRESS_PROPERTY. Příklad:

```
(MQEnvironment.properties).put(CMQC.LOCAL_ADDRESS_PROPERTY,
                               "192.0.2.0(2000,3000)");
```

- Když můžete sestavit objekt MQQueueManager , můžete předat hašovací tabulku vlastností obsahující LOCAL_ADDRESS_PROPERTY hodnotou "192.0.2.0(2000,3000)".

V každém z těchto příkladů, když se aplikace později připojí ke správci front, se aplikace připojí k lokální adrese IP a číslu portu v rozsahu 192.0.2.0(2000) na 192.0.2.0(3000).

V systému s více než jedním síťovým rozhraním můžete také použít pole Nastavení localAddressnebo vlastnost prostředí CMQC.LOCAL_ADDRESS_PROPERTY, chcete-li určit, které síťové rozhraní musí být použito pro připojení.

Pokud omezíte rozsah portů, může dojít k chybám připojení. Dojde-li k chybě, dojde k výjimce MQException obsahující kód příčiny IBM MQ MQRC_Q_MGR_NOT_AVAILABLE a následující zpráva:

```
Socket connection attempt refused due to LOCAL_ADDRESS_PROPERTY restrictions
```

Může se vyskytnout chyba, pokud jsou použity všechny porty v uvedeném rozsahu, nebo pokud zadaná adresa IP, název hostitele nebo číslo portu nejsou platné (například záporné číslo portu).

Přístup k frontám, tématům a procesům v produktu IBM MQ classes for Java

Chcete-li přistupovat k frontám, tématům a procesům, použijte metody třídy MQQueueManager . MQOD (struktura deskriptoru objektu) je sbalena do parametrů těchto metod.

Fronty

Chcete-li otevřít frontu, můžete použít metodu accessQueue třídy MQQueueManager . Například u správce front s názvem queueManagerpoužijte následující kód:

```
MQQueue queue = queueManager.accessQueue("qName",CMQC.MQOO_OUTPUT);
```

Metoda accessQueue vrací nový objekt třídy MQQueue.

Až skončíte s používáním fronty, použijte metodu close () k její zavření, jako v následujícím příkladu:

```
queue.close();
```

Frontu lze vytvořit také pomocí konstruktoru MQQueue. Parametry jsou přesně stejné jako u metody accessQueue spolu s parametrem správce front. Příklad:

```
MQQueue queue = new MQQueue(queueManager,
                            "qName",
                            CMQC.MQOO_OUTPUT,
                            "qMgrName",
                            "dynamicQName",
                            "altUserID");
```

Při vytváření front můžete zadat řadu voleb. Podrobnosti o těchto údajích naleznete v dokumentu Class.com.ibm.mq.MQQueue. Při vytváření objektu fronty tímto způsobem lze zapisovat do vlastních podtříd fronty MQQueue.

Témata

Podobně je možné otevřít téma pomocí metody accessTopic třídy MQQueueManager . Například ve správci front s názvem queueManagerpoužijte k vytvoření odběratele a vydavatele následující kód:

```
MQTopic subscriber =
      queueManager.accessTopic("TOPICSTRING","TOPICNAME",
      CMQC.MQTOPIC_OPEN_AS_SUBSCRIPTION, CMQC.MQSO_CREATE);
```

```
MQTopic publisher =
      queueManager.accessTopic("TOPICSTRING","TOPICNAME",
      CMQC.MQTOPIC_OPEN_AS_PUBLICATION, CMQC.MQOO_OUTPUT);
```

Po dokončení používání daného tématu ji zavřete pomocí metody close ().

Rovněž můžete vytvořit téma pomocí konstruktoru MQTopic. Parametry jsou přesně stejné jako u metody accessTopic spolu s přidáním parametru správce front. Příklad:

```
MQTopic subscriber = new
      MQTopic(queueManager,"TOPICSTRING","TOPICNAME",
      CMQC.MQTOPIC_OPEN_AS_SUBSCRIPTION, CMQC.MQSO_CREATE);
```

Při vytváření témat můžete zadat řadu voleb. Podrobné informace o těchto tématech naleznete v tématu Třída com.ibm.mq.MQTopic. Při vytváření objektu tématu v tomto způsobu lze psát vlastní podtřídy MQTopic.

Téma musí být otevřeno buď pro publikování, nebo pro odběr. Třída MQQueueManager má osm metod accessTopic a třída Topic má osm konstruktorů. V každém případě čtyři z nich mají parametr **destination** a čtyři mají parametr **subscriptionName** (včetně dvou, které mají obě). Ty lze použít pouze k otevření tématu pro odběry. Tyto dvě zbývající metody mají parametr **openAs** a lze je otevřít buď pro publikování, nebo pro odběr, v závislosti na hodnotě parametru **openAs** .

Chcete-li vytvořit téma jako trvalý odběratel, použijte buď metodu accessTopic třídy MQQueueManager , nebo konstruktor MQTopic, který přijímá název odběru, a v obou případech nastavte položku CMQC.MQSO_DURABLE .

Procesy

Chcete-li přistupovat k procesu, použijte metodu accessProcess objektu MQQueueManager. Například ve správci front s názvem queueManagervytvořte objekt MQProcess pomocí následujícího kódu:

```
MQProcess process =
queueManager.accessProcess("PROCESSNAME",
CMQC.MQOO_FAIL_IF_QUIESCING);
```

Chcete-li přistupovat k procesu, použijte metodu accessProcess objektu MQQueueManager.

Metoda accessProcess vrací nový objekt třídy MQProcess.

Když jste dokončili použití objektu procesu, použijte metodu close () k zavření, jako v následujícím příkladu:

```
process.close();
```

Proces můžete také vytvořit pomocí konstruktoru MQProcess. Parametry jsou přesně stejné jako u metody accessProcess spolu s přidáním parametru správce front. Příklad:

```
MQProcess process =
      new MQProcess(queueManager,"PROCESSNAME",
      CMQC.MQOO_FAIL_IF_QUIESCING);
```

Konstrukce objektu procesu tímto způsobem umožňuje zápis do vlastních podtříd procesu MQProcess.

Zpracování zpráv v produktu IBM MQ classes for Java

Zprávy jsou reprezentovány třídou MQMessage. Pomocí metod třídy MQDestination, která má podtřídy MQQueue a MQTopic, jste umístili a získali zprávy pomocí metod třídy MQDestination.

Vložení zpráv do front nebo témat pomocí metody put () třídy MQDestination. Zprávy získáte z front nebo témat pomocí metody get () třídy MQDestination. Na rozdíl od procedurálního rozhraní, kde příkazy MQPUT a MQGET vkládá a získává pole bajtů, uvádí programovací jazyk Java třídu MQMessage a získává instance třídy MQMessage. Třída MQMessage zapouzdřuje vyrovnávací paměť dat, která obsahuje skutečná data zprávy, spolu se všemi parametry MQMD (deskriptor zprávy) a vlastnostmi zpráv, které tuto zprávu popisují.

Chcete-li vytvořit novou zprávu, vytvořte novou instanci třídy MQMessage a pomocí metod writeXXX umístěte data do vyrovnávací paměti zpráv.

Když se vytvoří nová instance zprávy, všechny parametry MQMD se automaticky nastaví na jejich výchozí hodnoty, jak je definováno v Počáteční hodnoty a deklarace jazyka pro MQMD. Metoda put () objektu MQDestination také používá instanci třídy voleb MQPutMessagejako parametr. Tato třída představuje strukturu MQPMO. Následující příklad vytvoří zprávu a vloží ji do fronty:

```
// Build a new message containing my age followed by my name
MQMessage myMessage = new MQMessage();
myMessage.writeInt(25);

String name = "Charlie Jordan";
myMessage.writeInt(name.length());
myMessage.writeBytes(name);

// Use the default put message options...
MQPutMessageOptions pmo = new MQPutMessageOptions();

// put the message
!queue.put(myMessage,pmo);
```

Metoda get () MQDestination vrací novou instanci MQMessage, která představuje zprávu, která byla právě převzata z fronty. Jako parametr má také instanci třídy voleb MQGetMessage. Tato třída představuje strukturu MQGMO.

Maximální velikost zprávy není třeba zadávat, protože metoda get () automaticky upravuje velikost vnitřní vyrovnávací paměti tak, aby se vešla do příchozí zprávy. K přístupu k datům ve vrácené zprávě použijte metody readXXX třídy MQMessage.

Následující příklad uvádí, jak získat zprávu z fronty:

```
// Get a message from the queue
MQMessage theMessage    = new MQMessage();
MQGetMessageOptions gmo = new MQGetMessageOptions();
queue.get(theMessage,gmo); // has default values
```

```
// Extract the message data
int age = theMessage.readInt();
int strLen = theMessage.readInt();
byte[] strData = new byte[strLen];
theMessage.readFully(strData,0,strLen);
String name = new String(strData,0);
```

Formát čísla, který metody čtení a zápisu používají, můžete změnit nastavením proměnné členа *encoding* .

Můžete změnit znakovou sadu, která má být použita pro čtení a zápis řetězců, nastavením proměnné členu *characterSet* .

Další informace viz “Třída MQMessage” na stránce 598.

Poznámka: Metoda writeUTF() MQMessage automaticky kóduje délku řetězce stejně jako bajty Unicode, které obsahuje. Když bude vaše zpráva přečtena jiným programem Java (pomocí readUTF()), je to nejjednodušší způsob, jak odeslat informace o řetězci.

Zlepšení výkonu přechodných zpráv v produktu IBM MQ classes for Java
Chcete-li zlepšit výkon při procházení zpráv nebo přijímání přechodných zpráv z aplikace klienta, můžete použít *čtení napřed*. Klientské aplikace používající příkaz MQGET nebo asynchronní spotřeba budou těžit ze zlepšení výkonu při procházení zpráv nebo při přijímání přechodných zpráv.

Obecné informace o prostředku pro dopředné čtení najdete v tématu .

V produktu IBM MQ classes for Javalze pomocí vlastností CMQC.MQSO_READ_AHEAD a CMQC.MQSO_NO_READ_AHEAD objektu MQQueue nebo MQTopic určit, zda mají spotřebitelé zpráv a prohlížečové prohlížeče povoleno používat dopředné čtení u daného objektu.

Asynchronní vkládání zpráv do produktu IBM MQ classes for Java
Chcete-li asynchronně vložit zprávu, nastavte MQPMO_ASYNC_RESPONSE.

Do front nebo témat vkládejte zprávy do front nebo témat pomocí metody put () třídy MQDestination. Chcete-li asynchronně vložit zprávu, tj. umožnění dokončení operace bez čekání na odezvu ze správce front, můžete nastavit MQPMO_ASYNC_RESPONSE v poli voleb volby MQPutMessage. Chcete-li určit úspěch nebo selhání asynchronních operací vložení, použijte volání funkce MQQueueManager.getAsync.

Publikování/odběr v produktu IBM MQ classes for Java
V produktu IBM MQ classes for Javaje téma představováno třídou MQTopic a vy ji publikujete pomocí metod MQTopic.put().

Obecné informace o publikování a odběru IBM MQ naleznete v tématu Systém zpráv publikování/odběru.

Zpracování záhlaví zpráv produktu IBM MQ pomocí produktu IBM MQ classes for Java
Poskytují se třídy Java představující různé typy záhlaví zprávy. Poskytují se také dvě pomocné třídy.

Rozhraní MQHeader

Objekty záhlaví jsou popsány rozhraním MQHeader, které poskytuje všestranně určené metody pro přístup k polím záhlaví a pro čtení a zápis obsahu zpráv. Každý typ záhlaví má svou vlastní třídu, která implementuje rozhraní MQHeader a přidává metody getter a setter pro jednotlivá pole. Například typ záhlaví MQRFH2 je reprezentován třídou MQRFH2 ; typ záhlaví MQDLH třídou MQDLH atd. Třídy záhlaví provádějí veškerá nezbytná konverze dat automaticky a mohou číst nebo zapisovat data v libovolném uvedeném numerickém kódování nebo znakové sadě (CCSID).

Důležité: Třídy záhlaví MQRFH2 považují zprávu za náhodný přístupový soubor, což znamená, že kurzor musí být umístěn na začátku příslušné zprávy. Před použitím interní třídy záhlaví zprávy jako MQRFH, MQRFH2, MQCIH, MQDEAD, MQIIH nebo MQXMIT se ujistěte, že aktualizujete pozici kurzoru zprávy na správné místo před předáním zprávy do třídy.

Pomocné třídy

Dvě nápovědné třídy, MQHeaderIterator a MQHeaderListpomáhají při čtení a dekódování (analyzování) obsahu záhlaví ve zprávách:

- Třída MQHeaderIterator funguje podobně jako java.util.Iterator. Pokud je ve zprávě více záhlaví, metoda next () vrací hodnotu true a metoda nextHeader() nebo next () vrací další objekt záhlaví.
- MQHeaderList funguje podobně jako java.util.List. Podobně jako MQHeaderIteratoranalyzuje obsah záhlaví, ale také vám umožňuje hledat konkrétní záhlaví, přidávat nová záhlaví, odebírat existující záhlaví, aktualizovat pole záhlaví a poté zapsat obsah záhlaví zpět do zprávy. Případně můžete vytvořit prázdný objekt MQHeaderList, poté jej naplnit instancemi záhlaví a zapsat jej do zprávy jednou nebo opakovaně.

Třídy MQHeaderIterator a MQHeaderList používají informace v MQHeaderRegistry , aby věděli, které třídy záhlaví produktu IBM MQ jsou přidruženy ke konkrétním typům zpráv a formátům. Objekt MQHeaderRegistry je konfigurován se znalostmi všech aktuálních formátů a typů záhlaví produktu IBM MQ a tříd implementace a můžete také registrovat své vlastní typy záhlaví.

Podpora je poskytována pro následující běžně používaná záhlaví IBM MQ

- MQRFH-Pravidla a záhlaví formátování
- MQRFH2 -Like MQRFH, který se používá k předávání zpráv do a ze zprostředkovatele zpráv patřícího do produktu IBM Integration Bus. Používá se také k uchování vlastností zprávy
- Most MQCIH- CICS
- MQDLH-Záhlaví nedoručených zpráv
- Záhlaví informací MQIIH- IMS
- MQRMH-záhlaví zprávy odkazu
- MQSAPH-záhlaví SAP
- MQWIH-Záhlaví informací o práci
- MQXQH-záhlaví přenosové fronty
- MQDH-záhlaví distribuce
- MQEPH-zapouzdřené záhlaví PCF

Také můžete definovat třídy reprezentující vaše vlastní záhlaví.

Chcete-li použít MQHeaderIterator k získání záhlaví RFH2 , nastavte buď MQGMO_PROPERTIES_FORCE_MQRFH2 ve volbách GetMessage, nebo nastavte vlastnost fronty PROPCTL na hodnotu FORCE.

Tisk všech záhlaví ve zprávě pomocí produktu IBM MQ classes for Java
V tomto příkladu instance třídy MQHeaderIterator analyzuje záhlaví ve frontě zpráv MQMessage, která byla přijata z fronty. Objekty MQHeader vrácené z metody nextHeader() zobrazují jejich strukturu a obsah při vyvolání metody toString .

```
import com.ibm.mq.MQMessage;
import com.ibm.mq.headers.MQHeader;
import com.ibm.mq.headers.MQHeaderIterator;
...
MQMessage message = ... // Message received from a queue.
MQHeaderIterator it = new MQHeaderIterator (message);

while (it.hasNext ())
{
    MQHeader header = it.nextHeader ();

    System.out.println ("Header type " + header.type () + ": " + header);
}
```

Přeskakuje se záhlaví ve zprávě pomocí IBM MQ classes for Java
V tomto příkladu funkce skipHeaders() funkce MQHeaderIterator umístí zprávu na čtení zprávy okamžitě za poslední záhlaví.

```
import com.ibm.mq.MQMessage;
import com.ibm.mq.headers.MQHeaderIterator;
...
MQMessage message = ... // Message received from a queue.
MQHeaderIterator it = new MQHeaderIterator (message);

it.skipHeaders ();
```

Vyhledání kódu příčiny ve zprávě s dead-letter pomocí produktu IBM MQ classes for Java
V tomto příkladu metoda čtení naplní objekt MQDLH čtením ze zprávy. Po operaci čtení je kurzor pro čtení zprávy umístěn okamžitě za obsah záhlaví MQDLH.

Zprávy ve frontě nedoručených zpráv správce front mají předponu záhlaví zablokovaných dopisů (MQDLH). Chcete-li se rozhodnout, jak tyto zprávy zpracovat-například k určení, zda mají být pokusy o opakované pokusy nebo jejich vyřazení, musí se podívat na kód příčiny obsažený v souboru MQDLH.

```
import com.ibm.mq.MQMessage;
import com.ibm.mq.headers.MQDLH;
...
MQMessage message = ... // Message received from the dead-letter queue.
MQDLH dlh = new MQDLH ();

dlh.read (message);

System.out.println ("Reason: " + dlh.getReason ());
```

Všechny třídy záhlaví také poskytují konstruktor convenience pro inicializaci přímo ze zprávy v jednom kroku. Takže kód v tomto příkladu by mohl být zjednodušen následujícím způsobem:

```
import com.ibm.mq.MQMessage;
import com.ibm.mq.headers.MQDLH;
...
MQMessage message = ... // Message received from the dead-letter queue.
MQDLH dlh = new MQDLH (message);

System.out.println ("Reason: " + dlh.getReason ());
```

Čtení a odebrání záhlaví ze zprávy s dead-letter pomocí produktu IBM MQ classes for Java
V tomto příkladu se MQDLH používá k odstranění záhlaví ze zprávy s dead-letter.

Aplikace pro zpracování smrtelného dopisu bude obvykle znovu odeslat zprávy, které byly zamítnuty, pokud jejich kód příčiny indikuje přechodnou chybu. Před opětovným odesláním zprávy musí být odebráno záhlaví MQDLH.

Tento příklad provádí následující kroky (viz komentáře v ukázkovém kódu):

1. Příkaz MQHeaderList přečte celou zprávu a každé záhlaví, které se v této zprávě objeví, se stane položkou v seznamu.
2. Zprávy nedoručených zpráv obsahují záhlaví MQDLH jako jejich první záhlaví, takže lze tuto položku najít v první položce seznamu záhlaví. Objekt MQDLH již byl naplněn daty ze zprávy, když je sestaven objekt MQHeaderList , takže není třeba volat jeho metodu čtení.
3. Kód příčiny je extrahován pomocí metody getReason() poskytnuté třídou MQDLH.
4. Kód příčiny byl zkontrolován a označuje, že je vhodné znovu odeslat zprávu. MQDLH se odebere pomocí metody remove () MQHeaderList .
5. Funkce MQHeaderList zapisuje svůj zbývající obsah do nového objektu zprávy. Nová zpráva nyní obsahuje vše v původní zprávě s výjimkou MQDLH a může být zapsána do fronty. Argument **true** konstruktoru a metodě write označuje, že tělo zprávy má být zadrženo v rámci MQHeaderLista že je znovu zapsáno.

6. Pole formátu v deskriptoru zprávy nové zprávy nyní obsahuje hodnotu, která byla dříve uvedena v poli formátu MQDLH. Data zprávy se shodují s číselným kódováním a sadou CCSID v deskriptoru zpráv.

```
import com.ibm.mq.MQMessage;
import com.ibm.mq.headers.MQDLH;
import com.ibm.mq.headers.MQHeaderList;
...
MQMessage message = ... // Message received from the dead-letter queue.
MQHeaderList list = new MQHeaderList (message, true); // Step 1.
MQDLH dlh = (MQDLH) list.get (0); // Step 2.
int reason = dlh.getReason (); // Step 3.
...
list.remove (dlh); // Step 4.

MQMessage newMessage = new MQMessage ();

list.write (newMessage, true); // Step 5.
newMessage.format = list.getFormat (); // Step 6.
```

Tisk obsahu zprávy pomocí produktu IBM MQ classes for Java
Tento příklad používá MQHeaderList k vytištění obsahu zprávy včetně jeho záhlaví.

Výstup obsahuje zobrazení veškerého obsahu hlavičky a také těla zprávy. Třída MQHeaderList dekóduje všechna záhlaví v jednom kroku, zatímco kroky MQHeaderIterator postupně procházejí procesem pod kontrolou aplikace. Tuto techniku můžete použít k poskytnutí jednoduchého nástroje ladění při zápisu aplikací produktu WebSphere MQ .

```
import com.ibm.mq.MQMessage;
import com.ibm.mq.headers.MQHeaderList;
...
MQMessage message = ... // Message received from a queue.

System.out.println (new MQHeaderList (message, true));
```

Tento příklad také vytiskne pole deskriptoru zpráv pomocí třídy MQMD. Metoda copyFrom() třídy com.ibm.mq.headers.MQMD naplní objekt záhlaví daty z polí deskriptoru zpráv MQMessage, nikoli čtením těla zprávy.

```
import com.ibm.mq.MQMessage;
import com.ibm.mq.headers.MQMD;
import com.ibm.mq.headers.MQHeaderList;
...
MQMessage message = ...
MQMD md = new MQMD ();
...
md.copyFrom (message);
System.out.println (md + "\n" + new MQHeaderList (message, true));
```

Vyhledání specifického typu záhlaví ve zprávě pomocí produktu IBM MQ classes for Java
Tento příklad používá metodu indexOf(String) objektu MQHeaderList k nalezení záhlaví MQRFH2 ve zprávě, je-li k dispozici.

```
import com.ibm.mq.MQMessage;
import com.ibm.mq.headers.MQHeaderList;
import com.ibm.mq.headers.MQRFH2;
...
MQMessage message = ...
MQHeaderList list = new MQHeaderList (message);
int index = list.indexOf ("MQRFH2");

if (index >= 0)
{
    MQRFH2 rfh = (MQRFH2) list.get (index);
        ...
}
```

Analýza záhlaví MQRFH2 pomocí produktu IBM MQ classes for Java
Tento příklad ukazuje, jak získat přístup ke známé hodnotě pole v pojmenované složce, pomocí třídy MQRFH2 .

Třída MQRFH2 poskytuje řadu způsobů, jak přistupovat nejen k polím v pevné části struktury, ale také k obsahu složek kódovaným XML, které jsou přenášeny v poli Data NameValue. Tento příklad ukazuje, jak získat přístup ke známé hodnotě pole ve jmenované složce-v této instanci, pole Rto ve složce jms, která představuje název fronty odpovědí ve zprávě MQ JMS .

```
MQRFH2 rfh = ...

String value = rfh.getStringFieldValue ("jms", "Rto");
```

Chcete-li zjistit obsah MQRFH2 (na rozdíl od požadavku přímo na specifická pole), můžete použít metodu getFolders k vrácení seznamu MQRFH2.Element, který představuje strukturu složky, která může obsahovat pole a další složky. Nastavení pole nebo složky na hodnotu null ji odebere ze struktury MQRFH2. Když manipulujete s obsahem datové složky NameValuetímto způsobem, pole StrucLength se automaticky aktualizuje odpovídajícím způsobem.

Čtení a zápis toků bajtů jiných než objekty MQMessage pomocí produktu IBM MQ classes for Java
Tyto příklady používají třídy záhlaví k analýze a manipulaci s obsahem záhlaví produktu IBM MQ v případě, že zdrojem dat není objekt MQMessage.

Třídy záhlaví můžete použít k analýze a manipulaci s obsahem záhlaví produktu IBM MQ i v případě, že je zdrojem dat něco jiného než objekt MQMessage. Rozhraní MQHeader, které je implementováno každou třídou záhlaví, poskytuje metody `int read (java.io.DataInput message, int encoding, int characterSet)` a `int write (java.io.DataOutput message, int encoding, int characterSet)`. Třída com.ibm.mq.MQMessage implementuje rozhraní java.io.DataInput a java.io.DataOutput . To znamená, že můžete použít dvě metody MQHeader ke čtení a zápisu obsahu MQMessage, přičemž bude potlačeno kódování a CCSID určený v deskriptoru zpráv. To je užitečné pro zprávy, které obsahují řetězec záhlaví v různých kódování.

Můžete také získat objekty DataInput a DataOutput z jiných toků dat, například pro toky souborů nebo soketů nebo bajtová pole, která jsou přenášeno ve zprávách produktu JMS . Třídy java.io.DataInputStream implementují třídy DataInput a třídy java.io.DataOutputStream implementují DataOutput. Tento příklad čte obsah záhlaví IBM MQ z bajtového pole:

```
import java.io.*;
import com.ibm.mq.headers.*;
...
byte [] bytes = ...
DataInput in = new DataInputStream (new ByteArrayInputStream (bytes));
MQHeaderIterator it = new MQHeaderIterator (in, CMQC.MQENC_NATIVE,
  CMQC.MQCCSI_DEFAULT);
```

Řádek, který spouští MQHeaderIterator , může být nahrazen pomocí

```
MQDLH dlh = new MQDLH (in, CMQC.MQENC_NATIVE, CMQC.MQCCSI_DEFAULT);
// or any other header type
```

Tento příklad zapisuje na bajtové pole pomocí proudu DataOutput:

```
MQHeader header = ... // Could be any header type
ByteArrayOutputStream out = new ByteArrayOutputStream ();

header.write (new DataOutputStream (out), CMQC.MQENC_NATIVE, CMQC.MQCCSI_DEFAULT);
byte [] bytes = out.toByteArray ();
```

Když pracujete s proudy tímto způsobem, buďte opatrní, abyste použili správné hodnoty pro kódování a argumenty characterSet . Při čtení záhlaví zadejte kódování a CCSID, se kterým byl původně zapsán bajtový obsah. Při zápisu záhlaví zadejte kódování a CCSID, které chcete vytvořit. Převod dat se provádí automaticky třídami záhlaví.

Vytváření tříd pro nové typy záhlaví pomocí produktu IBM MQ classes for Java
Můžete vytvořit třídy Java pro typy záhlaví, které nejsou dodány s produktem IBM MQ classes for Java.

Chcete-li přidat třídu Java představující nový typ záhlaví, který lze použít stejným způsobem, jako každá třída záhlaví dodávaná s produktem IBM MQ classes for Java, vytvoříte třídu, která implementuje rozhraní MQHeader. Nejjednodušším přístupem je rozšířit třídu com.ibm.mq.headers.impl.Header . Tento příklad vytvoří plně funkční třídu reprezentující strukturu záhlaví MQTM. Nemusíte přidávat jednotlivé metody getter a setter pro každé pole, ale je to užitečné pohodlí pro uživatele třídy záhlaví. Generické metody getValue a setValue , které přijmou řetězec pro název pole, budou pracovat pro všechna pole definovaná v typu záhlaví. Zděděné metody čtení, zápisu a velikosti umožní instancím nového typu záhlaví, které mají být čteny a zapsány, a vypočítá velikost záhlaví správně na základě definice pole. Definice typu se vytvoří právě jednou, ale vytvoří se mnoho instancí této třídy záhlaví. Chcete-li zpřístupnit novou definici záhlaví pro dekódování pomocí tříd MQHeaderIterator nebo MQHeaderList , zaregistrujete ji pomocí MQHeaderRegistry. Všimněte si však, že třída záhlaví MQTM je ve skutečnosti již v tomto balíku poskytnuta a registrována ve výchozím registru.

```
import com.ibm.mq.headers.impl.Header;
import com.ibm.mq.headers.impl.HeaderField;
import com.ibm.mq.headers.CMQC;

public class MQTM extends Header {
    final static HeaderType TYPE = new HeaderType ("MQTM");
    final static HeaderField StrucId = TYPE.addMQChar ("StrucId", CMQC.MQTM_STRUC_ID);
    final static HeaderField Version = TYPE.addMQLong ("Version", CMQC.MQTM_VERSION_1);
    final static HeaderField QName = TYPE.addMQChar ("QName", CMQC.MQ_Q_NAME_LENGTH);
    final static HeaderField ProcessName = TYPE.addMQChar ("ProcessName",
      CMQC.MQ_PROCESS_NAME_LENGTH);
    final static HeaderField TriggerData = TYPE.addMQChar ("TriggerData",
      CMQC.MQ_TRIGGER_DATA_LENGTH);
    final static HeaderField ApplType = TYPE.addMQLong ("ApplType");
    final static HeaderField ApplId = TYPE.addMQChar ("ApplId", 256);
    final static HeaderField EnvData = TYPE.addMQChar ("EnvData", 128);
    final static HeaderField UserData = TYPE.addMQChar ("UserData", 128);

    protected MQTM (HeaderType type){
        super (type);
    }
    public String getStrucId () {
        return getStringValue (StrucId);
    }
    public int getVersion () {
        return getIntValue (Version);
    }
    public String getQName () {
        return getStringValue (QName);
    }
    public void setQName (String value) {
        setStringValue (QName, value);
    }
    // ...Add convenience getters and setters for remaining fields in the same way.
}
```

Práce se zprávami PCF s IBM MQ classes for Java
Třídy Java jsou poskytovány pro vytváření a analýzu zpráv strukturovaných PCF a pro usnadnění odesílání požadavků PCF a shromažďování odpovědí PCF.

Třídy PCFMessage & MQCFGR představují pole struktur parametrů PCF. Poskytují pohodlnější metody pro přidávání a načítání parametrů PCF.

Struktury parametrů PCF jsou reprezentovány třídami MQCFH, MQCFIN, MQCFIN64, MQCFST, MQCFBS, MQCFIL, MQCFIL64 MQCFSL a MQCFGR. Tato sdílená základní operační rozhraní:

- Metody pro čtení a zápis obsahu zpráv: read (), write () a size ()
- Metody pro manipulaci s parametry: getValue (), setValue (), getParameter () a další
- Metoda výčtového nástroje.nextParameter (), která analyzuje obsah PCF ve zprávě MQMessage

Parametr filtru PCF se používá v dotazových příkazech k poskytnutí funkce filtru. Zapouzdřeno v následujících třídách:

- MQCFIF-celočíselný filtr
- MQCFSF-filtr řetězců
- MQCFBF-filtr bajtů

Jsou poskytnuty dvě třídy agenta, PCFAgent a PCFMessageAgent pro správu připojení ke správci front, frontu příkazového serveru a přidruženou frontu odpovědí. PCFMessageAgent rozšiřuje PCFAgent a měl by se normálně používat v předvolbě. Třída PCFMessageAgent převádí přijaté MQMessages a předává je zpět volajícímu jako pole PCFMessage. Modul PCFAgent vrací pole zpráv MQMessages, které je třeba před použitím analyzovat.

Práce s vlastnostmi zpráv v produktu IBM MQ classes for Java

Volání funkcí ke zpracování obslužných rutin zpráv nemají v produktu IBM MQ classes for Javažádný ekvivalent. Chcete-li nastavit, vrátit nebo odstranit vlastnosti obslužné rutiny zpráv, použijte metody třídy MQMessage.

Obecné informace o vlastnostech zprávy viz “Názvy vlastností” na stránce 22.

V přístupu produktu IBM MQ classes for Java ke zprávám se používá třída MQMessage. Popisovače zpráv nejsou proto v prostředí Java poskytnuty a neexistuje žádný ekvivalent volání funkce IBM MQ MQCRTMH, MQDLTMH, MQMHBUF a MQBUFMH

Chcete-li nastavit vlastnosti zpracování zpráv v procedurálním rozhraní, použijte volání MQSETMP volání. V produktu IBM MQ classes for Javapoužijte příslušnou metodu třídy MQMessage:

- Vlastnost setBoolean
- Vlastnost setByte
- Vlastnost setBytes
- Vlastnost setShort
- Vlastnost setInt
- setInt2Property
- setInt4Property
- setInt8Property
- Vlastnost setLong
- Vlastnost setFloat
- Vlastnost setDouble
- Vlastnost setString
- Vlastnost setObject

Ty se někdy souhrnně označují jako metody *set*property* .

Chcete-li vrátit hodnotu vlastností manipulátoru zprávy v procedurálním rozhraní, použijte volání MQINQMP volání. V produktu IBM MQ classes for Javapoužijte příslušnou metodu třídy MQMessage:

- Vlastnost getBoolean
- Vlastnost getByte
- Vlastnost getBytes
- Vlastnost getShort
- Vlastnost getInt
- getInt2Property
- getInt4Property
- getInt8Property
- Vlastnost getLong
- Vlastnost getFloat

- Vlastnost getDouble
- Vlastnost getString
- Vlastnost getObject

Tyto metody jsou někdy souhrnně označovány jako metody *get*property* .

Chcete-li odstranit hodnotu vlastností zpracování zpráv v procedurálním rozhraní, použijte příkaz MQDLTMP volání. V produktu IBM MQ classes for Javapoužijte metodu deleteProperty třídy MQMessage.

Ošetření chyb v produktu IBM MQ classes for Java

Zpracovat chyby vzniklé z IBM MQ classes for Java pomocí bloků Java `try` a `catch` .

Metody v rozhraní produktu Java nevracejí kód dokončení a kód příčiny. Místo toho vyhodí výjimku pokaždé, když kód dokončení a kód příčiny, které jsou výsledkem volání IBM MQ , nejsou obě nula. To zjednodušuje logiku programu, takže nemusíte kontrolovat návratové kódy po každém volání do IBM MQ. Můžete se rozhodnout, ve kterých bodech ve vašem programu se chcete vypořádat s možností selhání. V těchto bodech můžete obklopovat kód pomocí bloků `try` a `catch` , jako v následujícím příkladu:

```
try {
    myQueue.put(messageA,putMessageOptionsA);
    myQueue.put(messageB,putMessageOptionsB);
}
catch (MQException ex) {
    // This block of code is only executed if one of
    // the two put methods gave rise to a non-zero
    // completion code or reason code.
    System.out.println("An error occurred during the put operation:" +
                           "CC = " + ex.completionCode +
                           "RC = " + ex.reasonCode);
    System.out.println("Cause exception:" + ex.getCause() );
}
```

Kódy příčiny volání příkazu IBM MQ , které jsou ohlášeny zpět ve Java výjimkách pro z/OS , jsou dokumentovány v kódu dokončení a kódu příčiny API.

Do protokolu se zapisují také výjimky, které jsou vyvoláné během spuštění aplikace IBM MQ classes for Java . Aplikace však může volat metodu MQException.logExclude(), a zabránit tak protokolování výjimek přidružených k určitému kódu příčiny. Možná to budete chtít provést v situacích, kdy očekáváte mnoho výjimek přidružených ke specifickému kódu příčiny, který má být vyhozen, a nechcete, aby byl protokol naplněn těmito výjimkami. Pokud se například vaše aplikace pokusí o získání zprávy z fronty pokaždé, když se opakuje okolo cyklu, a u většiny z těchto pokusů neočekáváte, že ve frontě není vhodná zpráva, můžete zabránit tomu, aby byly zaprotokolovány výjimky přidružené k kódu příčiny MQRC_NO_MSG_AVAILABLE. Pokud aplikace již dříve zaprotokolovala výjimky přidružené ke specifickému kódu příčiny, může povolit, aby tyto výjimky byly protokolovány znovu voláním metody MQException.logInclude().

Někdy kód příčiny nepředává všechny podrobnosti přidružené k chybě. Pro každou výjimku, která je vygenerována, by aplikace měla zkontrolovat připojenou výjimku. Samotná připojená výjimka může mít jinou propojené výjimky a propojené výjimky tvoří řetězec vedoucí zpět k původnímu základnímu problému. Propojená výjimka je implementována použitím mechanizmu výjimek v řetězové výjimce třídy java.lang.Throwable a aplikace obdrží připojenou výjimku voláním metody Throwable.getCause(). Z výjimky, která je instancí MQException, načte MQException.getCause() základní instanci com.ibm.mq.jmqi.JmqiExceptiona getCause z této výjimky načte základní java.lang.Exception , která způsobila chybu.

Získání a nastavení hodnot atributu v produktu IBM MQ classes for Java

Metody getXXX() a setXXX() jsou poskytovány pro mnoho obecných atributů. K ostatním lze přistupovat pomocí generických metod zjišťování () a set ().

U mnoha obecných atributů třídy MQManagedObject, MQDestination, MQQueue, MQTopic, MQProcess a MQQueueManager obsahují metody getXXX() a setXXX(). Tyto metody umožňují získat a nastavit hodnoty atributů. Všimněte si, že pro objekty MQDestination, MQQueue a MQTopic pracují metody pouze tehdy, zadáte-li při otevření objektu příslušné parametry dotazu a nastavení.

Pro méně běžné atributy jsou všechny třídy MQQueueManager, MQDestination, MQQueue, MQTopic a MQProcess všechny dědit ze třídy s názvem MQManagedObject. Tato třída definuje dotazová rozhraní () a set ().

Když vytváříte nový objekt správce front pomocí operátoru *new* , je automaticky otevřen pro zjišťování. Použijete-li metodu accessProcess() k přístupu k objektu procesu, je tento objekt automaticky otevřen pro zjišťování. Použijete-li metodu accessQueue() pro přístup k objektu fronty, nebude tento objekt automaticky otevřen pro dotazování nebo nastavení operací. Důvodem je to, že přidání těchto voleb automaticky může způsobit problémy s některými typy vzdálených front. Chcete-li ve frontě použít metody zjišťování, nastavení, getXXXa setXXX , musíte v parametru openOptions metody accessQueue() určit příslušné parametry dotazu a nastavit příznaky. To samé platí pro cílové objekty a objekty témat.

Metody dotazování a nastavení mají tři parametry:

- pole selektorů
- Pole intAttrs
- pole charAttrs

Nepotřebujete parametry SelectorCount, IntAttrCount a CharAttrLength, které se nacházejí v MQINQ, protože délka pole v Java je vždy známá. Následující příklad uvádí, jak provést dotaz na frontu:

```
// inquire on a queue
final static int MQIA_DEF_PRIORITY = 6;
final static int MQCA_Q_DESC = 2013;
final static int MQ_Q_DESC_LENGTH = 64;

int[] selectors  = new int[2];
int[] intAttrs   = new int[1];
byte[] charAttrs = new byte[MQ_Q_DESC_LENGTH]

selectors[0] = MQIA_DEF_PRIORITY;
selectors[1] = MQCA_Q_DESC;

queue.inquire(selectors,intAttrs,charAttrs);

System.out.println("Default Priority = " + intAttrs[0]);
System.out.println("Description : " + new String(charAttrs,0));
```

Vícevláknové programy v produktu Java

Běhové prostředí produktu Java je ve své podstatě vícevláknové. Produkt IBM MQ classes for Java umožňuje sdílení objektu správce front více podprocesy, ale zajišťuje synchronizaci všech přístupů k cílovému správci front.

Vícevláknové programy se mohou v produktu Javavyhnout. Zvažte jednoduchý program, který se připojí ke správci front a otevře frontu při spuštění. Program zobrazí na obrazovce jediné tlačítko. Když uživatel klepne na toto tlačítko, program načte zprávu z fronty.

Běhové prostředí produktu Java je ve své podstatě vícevláknové. Proto se inicializace vaší aplikace vyskytuje v jednom podprocesu a kód, který se provádí v odpovědi na stisknutí tlačítka, se provádí v samostatném podprocesu (podproces uživatelského rozhraní).

S IBM MQ MQI clientzaloženým na C by tento problém způsobil problém, protože existují určitá omezení sdílení popisovačů více podprocesy. Produkt IBM MQ classes for Java uvolní toto omezení, což umožňuje sdílení objektu správce front (a jeho přidružené fronty, objektů témat a procesů) více podprocesy.

Implementace produktu IBM MQ classes for Java zajišťuje, že pro konkrétní připojení (instance objektuMQQueueManager) je veškerý přístup k cílovému správci front IBM MQ synchronizován. Podproces, který chce vydat volání správci front, je blokován, dokud nejsou dokončena všechna ostatní volání v průběhu tohoto připojení. Pokud vyžadujete souběžný přístup ke stejnému správci front z více podprocesů ve svém programu, vytvořte nový objekt MQQueueManager pro každý podproces, který vyžaduje souběžný přístup. (Toto je ekvivalent k zadání samostatného volání MQCONN pro každé vlákno.)

Poznámka: Instance třídy `com.ibm.mq.MQGetMessageOptions` nesmí být sdíleny mezi podprocesy, které současně vyžadují zprávy. Instance této třídy se aktualizují daty během odpovídající žádosti MQGET,

což může mít za následek neočekávané důsledky, pokud více podprocesů souběžně pracuje na stejné instanci objektu.

Použití kanálů kanálů v produktu IBM MQ classes for Java

Přehled způsobů použití kanálů v aplikaci pomocí produktu IBM MQ classes for Java.

Následující témata popisují způsob zápisu uživatelské procedury kanálu do produktu Java, způsobu jeho přiřazení a způsobu předávání dat do tohoto kanálu. Pak popisují, jak se používají uživatelské procedury kanálu napsané v C a jak používat sled uživatelských procedur.

Vaše aplikace musí mít nastaveno správné oprávnění zabezpečení pro načtení třídy uživatelských procedur kanálu.

Vytvoření uživatelské procedury kanálu v produktu IBM MQ classes for Java

Vlastní uživatelské procedury kanálu můžete zajistit definováním třídy Java , která implementuje příslušné rozhraní.

Chcete-li implementovat uživatelskou proceduru, definujte novou třídu Java , která implementuje příslušné rozhraní. Tři výstupní rozhraní jsou definována v balíku com.ibm.mq.exits :

- WMQSendExit
- WMQReceiveExit
- WMQSecurityExit

Poznámka: Uživatelské procedury kanálu jsou podporovány pouze pro připojení klienta; nejsou podporovány pro připojení vazeb. Nemůžete použít uživatelskou proceduru kanálu Java mimo produkt IBM MQ classes for Java, například pokud používáte klientskou aplikaci napsanou v C.

Jakékoli šifrování TLS definované pro připojení se provádí *po* vyvolání a byly vyvolány uživatelské procedury pro zabezpečení odeslání a zabezpečení. Podobně se dešifrování provádí *před* a vyvoláno ukončení zabezpečení.

Následující ukázka definuje třídu, která implementuje všechna tři rozhraní:

```
public class MyMQExits implements
WMQSendExit, WMQReceiveExit, WMQSecurityExit {
    // Default constructor
  public MyMQExits(){
  }
    // This method comes from the send exit interface
  public ByteBuffer channelSendExit(
MQCXP channelExitParms,
                                  MQCD channelDefinition,
                                  ByteBuffer agentBuffer)
  {
    // Fill in the body of the send exit here
  }
    // This method comes from the receive exit interface
  public ByteBuffer channelReceiveExit(
MQCXP channelExitParms,
                                     MQCD channelDefinition,
                                     ByteBuffer agentBuffer)
  {
    // Fill in the body of the receive exit here
  }
    // This method comes from the security exit interface
  public ByteBuffer channelSecurityExit(
MQCXP channelExitParms,
                                      MQCD channelDefinition,
                                      ByteBuffer agentBuffer)
  {
    // Fill in the body of the security exit here
  }
}
```

Každé uživatelské proceduře je předán objekt MQCXP a objekt MQCD. Tyto objekty reprezentují struktury MQCXP a MQCD definované v procedurálním rozhraní.

Každá třída ukončení, kterou napíšete, musí mít konstruktor. Může se jednat buď o výchozí konstruktor, nebo o takový, který bude mít řetězcový argument. Pokud je řetězec zapotřebí, budou data uživatele předána do třídy ukončení při jejím vytvoření. Pokud třída uživatelské procedury obsahuje jak výchozí konstruktor, tak konstruktor s jedním argumentem, má konstruktor s jedním argumentem prioritu.

V případě uživatelských procedur pro odesílání a zabezpečení musí návratový kód vracet data, která chcete odeslat na server. Pro uživatelskou proceduru příjmu musí návratový kód vracet upravená data, která chcete interpretovat jako IBM MQ .

Nejjednodušším možným výstupním tělem je:

```
{ return agentBuffer; }
```

Nezavírejte správce front v rámci uživatelské procedury kanálu.

Použití existujících tříd uživatelské procedury kanálu

Ve verzích produktu IBM MQ starších než 7.0byla tato uživatelská procedura použita pomocí rozhraní MQSendExit, MQReceiveExita MQSecurityExit, jak je uvedeno v následujícím příkladu. Tato metoda zůstává platná, ale nová metoda je upřednostňována pro zlepšení funkčnosti a výkonu.

```
public class MyMQExits implements MQSendExit, MQReceiveExit, MQSecurityExit {
    // Default constructor
  public MyMQExits(){
  }
    // This method comes from the send exit
  public byte[] sendExit(MQChannelExit channelExitParms,
                         MQChannelDefinition channelDefParms,
                         byte agentBuffer[])
  {
    // Fill in the body of the send exit here
  }
    // This method comes from the receive exit
  public byte[] receiveExit(MQChannelExit channelExitParms,
                            MQChannelDefinition channelDefParms,
                            byte agentBuffer[])
  {
    // Fill in the body of the receive exit here
  }
    // This method comes from the security exit
  public byte[] securityExit(MQChannelExit channelExitParms,
                             MQChannelDefinition channelDefParms,
                             byte agentBuffer[])
  {
    // Fill in the body of the security exit here
  }
}
```

Přiřazení ukončení kanálu v produktu IBM MQ classes for Java
Uživatelská procedura kanálu můžete přiřadit pomocí produktu IBM MQ classes for Java.

V produktu IBM MQ classes for Javaneexistuje žádný přímý ekvivalent k kanálu IBM MQ . Uživatelské procedury kanálu jsou přiřazeny k objektu MQQueueManager. Definujete-li například třídu, která implementuje rozhraní WMQSecurityExit , může aplikace použít uživatelskou proceduru zabezpečení jedním ze čtyř způsobů:

- Přiřazením instance třídy do pole MQEnvironment.channelSecurityExit před vytvořením objektu MQQueueManager
- Nastavením pole MQEnvironment.channelSecurityExit na řetězec reprezentující třídu uživatelské procedury zabezpečení před vytvořením objektu MQQueueManager .
- Vytvořením dvojice klíč/hodnota v hašovací tabulce vlastností předané aplikaci MQQueueManager s klíčem CMQC.SECURITY_EXIT_PROPERTY .
- Použití tabulky CCDT (Client Channel Definition table)

Každá uživatelská procedura přiřazená nastavením pole MQEnvironment.channelSecurityExit na řetězec, vytvoření dvojice klíč/hodnota v tabulce vlastností vlastností nebo použití tabulky CCDT, musí být napsána s výchozím konstruktorem. Ukončení přiřazené jako instance třídy nepotřebuje výchozí konstruktor, v závislosti na aplikaci.

Aplikace může podobným způsobem použít odeslání nebo přijetí pro přijetí. Například následující fragment kódu ukazuje, jak používat uživatelské procedury zabezpečení, odeslání a přijetí implementované ve třídě MyMQExits, která byla definována dříve pomocí prostředí MQEnvironment:

```
MyMQExits myexits = new MyMQExits();
MQEnvironment.channelSecurityExit = myexits;
MQEnvironment.channelSendExit = myexits;
MQEnvironment.channelReceiveExit = myexits;
:
MQQueueManager jupiter = new MQQueueManager("JUPITER");
```

Je-li k přiřazení uživatelské procedury kanálu použita více než jedna metoda, bude mít pořadí přednosti následující:

1. Je-li adresa URL tabulky CCDT předána do správce MQQueueManager, obsah kanálu CCDT určuje ukončení kanálu, které má být použito, a všechny definice ukončení v prostředí MQEnvironment nebo hašovací tabulka vlastností jsou ignorovány.
2. Není-li předána žádná adresa URL tabulky CCDT, dojde ke sloučení definic ukončení z prostředí MQEnvironment a hašovací tabulky.
   - Je-li definován stejný typ ukončení v rozhraní MQEnvironment i v tabulce hashtable, použije se definice v tabulce hashtable.
   - Jsou-li zadány stejné staré a nové typy ukončení (například pole sendExit , které lze použít pouze pro typ ukončení použitý ve verzích starších než IBM WebSphere MQ 7.0, a pole Konec channelSend, které lze použít pro jakoukoli uživatelskou proceduru pro odeslání zprávy), použije se nová uživatelská procedura (channelSendExit) namísto staré uživatelské procedury.

Pokud jste deklarovali uživatelskou proceduru kanálu jako řetězec, musíte produkt IBM MQ povolit, aby nalezl ukončovací program kanálu. Můžete tak učinit různými způsoby v závislosti na prostředí, ve kterém je aplikace spuštěna, a na tom, jak jsou uživatelské programy kanálu zabaleny.

- Pro aplikaci spuštěnou na aplikačním serveru musíte uložit soubory do adresáře zobrazeného v produktu Tabulka 58 na stránce 353 nebo zabalené do souborů JAR odkazovaných produktem **exitClasspath**.
- V případě aplikace, která není spuštěna na aplikačním serveru, platí následující pravidla:
  - Pokud jsou vaše třídy ukončení kanálu zabaleny do samostatných souborů JAR, tyto soubory JAR musí být obsaženy v produktu **exitClasspath**.
  - Nejsou-li třídy uživatelské procedury kanálu zabaleny v souborech JAR, lze soubory tříd uložit do adresáře zobrazeného v produktu Tabulka 58 na stránce 353 nebo do libovolného adresáře v cestě ke třídám systému JVM nebo **exitClasspath**.

Vlastnost **exitClasspath** může být zadána čtyřmi způsoby. V zájmu priority jsou tyto způsoby následující:

1. Systémová vlastnost com.ibm.mq.exitClasspath (definovaná na příkazovém řádku pomocí volby `-D`).
2. Stanza exitPath souboru `mqclient.ini`
3. Záznam hašovací tabulky s klíčem CMQC.EXIT_CLASSPATH_PROPERTY
4. Proměnná MQEnvironment **exitClasspath**

Oddělte více cest pomocí znaku java.io.File.pathSeparator .

Tabulka 58. Adresář pro uživatelské programy kanálu	
Platforma	**Adresář**
AIX , HP-UX, Linux a Solaris	`/var/mqm/exits` (programy s 32bitovým ukončovacím programem kanálu) `/var/mqm/exits64` (programy s 64bitovým ukončovacím programem)
Windows	*instalační_dat_adr*\exits

Poznámka: *instalační_dat_adr* je adresář, který jste vybrali pro datové soubory produktu IBM MQ během instalace. Standardní adresář je `C:\ProgramData\IBM\MQ`.

Předání dat do kanálů kanálu v produktu IBM MQ classes for Java
Můžete předávat data do ukončení kanálu a vracet data z kanálů z kanálů do vaší aplikace.

Parametr agentBuffer

Pro uživatelskou proceduru odeslání zprávy obsahuje parametr *agentBuffer* data, která se mají odeslat. Pro uživatelskou proceduru pro přijetí zprávy nebo ukončení zabezpečení obsahuje parametr *agentBuffer* data, která právě byla přijata. Nepotřebujete mít parametr délky, protože výraz agentBuffer.limit () označuje délku pole.

V případě uživatelských procedur pro odesílání a zabezpečení musí návratový kód vracet data, která chcete odeslat na server. Pro uživatelskou proceduru příjmu musí návratový kód vracet upravená data, která chcete interpretovat jako IBM MQ .

Nejjednodušším možným výstupním tělem je:

```
{ return agentBuffer; }
```

Uživatelské procedury kanálu jsou volány s vyrovnávací pamětí, která má záložní pole. Pro nejlepší výkon by měla uživatelská procedura vrátit vyrovnávací paměť s záložním polem.

Data uživatele

Pokud se aplikace připojí ke správci front nastavením volby channelSecurityExit, channelSendExit nebo channelReceiveExit, lze do příslušné třídy uživatelských procedur kanálu předat 32 bajtů uživatelských dat pomocí polí channelSecurityExitUserData, channelSendExitUserData, nebo channelReceiveExitUserData. Tato uživatelská data jsou k dispozici pro třídu uživatelské procedury kanálu, ale je aktualizována při každém zavolání uživatelské procedury. Jakékoli změny provedené v uživatelských datech v uživatelské proceduře kanálu budou proto ztraceny. Chcete-li provádět trvalé změny dat v uživatelské proceduře kanálu, použijte oblast MQCXP exitUser. Data v tomto poli se udržují mezi vyvoláními ukončení.

Pokud aplikace nastaví securityExit, sendExitnebo receiveExit, nelze do těchto tříd uživatelské procedury kanálu předat žádná uživatelská data.

Pokud aplikace používá tabulku definic kanálů klienta (CCDT) k připojení ke správci front, budou veškerá uživatelská data zadaná v definici kanálu připojení klienta předávána třídám uživatelské procedury kanálu při jejich volání. Další informace o použití tabulky definic kanálů klienta viz “Použití tabulky definic kanálů klienta s IBM MQ classes for Java” na stránce 336.

Použití uživatelských procedur kanálu nezapsaných v Java s IBM MQ classes for Java
Způsob použití ukončovacích programů kanálu napsaných v jazyce C z aplikace Java .

V produktu IBM WebSphere MQ 7.0můžete zadat název uživatelského programu kanálu zapsaného v jazyce C jako řetězec předaný do polí channelSecurityExit, channelSendExit nebo channelReceivev objektu MQEnvironment nebo vlastností Hashtable vlastností. Nelze však použít uživatelskou proceduru kanálu napsanou v Java v aplikaci napsanou v jiném jazyce.

Zadejte jméno ukončovacího programu ve formátu `library(function)` a ujistěte se, že umístění uživatelského programu je uvedeno, jak je popsáno v Cesta k ukončení.

Použití externích tříd ukončení

Ve verzích starších než IBM WebSphere MQ 7.0byly k dispozici tři třídy, které vám umožňují používat uživatelské procedury kanálu zapsané v jiných jazycích než Java:

- MQExternalSecurityExit, který implementuje rozhraní MQSecurityExit
- MQExternalSendExit, který implementuje rozhraní MQSendExit
- MQExternalReceiveExit, který implementuje rozhraní MQReceiveExit

Použití těchto tříd zůstává v platnosti, ale dává přednost nové metodě.

Chcete-li použít uživatelskou proceduru pro zabezpečení zprávy, která není napsána v produktu Java, aplikace nejprve musela vytvořit objekt Ukončení MQExternalSecurity. Uvedená aplikace, jako parametry v konstruktoru MQExternalSecurityExit, název knihovny obsahující uživatelskou proceduru pro zabezpečení zprávy, jméno vstupního bodu pro uživatelskou proceduru zabezpečení a uživatelská data, která mají být předána uživatelské proceduře pro zabezpečení při volání. Ukončovací programy kanálu, které nejsou zapsány v produktu Java , byly uloženy do adresáře zobrazeného vTabulka 58 na stránce 353.

Použití posloupnosti odeslání nebo přijetí kanálu v produktu IBM MQ classes for Java
Aplikace produktu IBM MQ classes for Java může použít posloupnost odeslaných nebo přijatých uživatelských procedur kanálu, které se spustí za sebou.

Chcete-li použít posloupnost uživatelských procedur pro odesílání, aplikace může vytvořit buď seznam, nebo řetězec obsahující uživatelské procedury odeslání. Je-li použit seznam, každý prvek seznamu může mít některou z následujících hodnot:

- Instance třídy definované uživatelem, která implementuje rozhraní WMQSendExit .
- Instance třídy definované uživatelem, která implementuje rozhraní MQSendExit (pro odeslání uživatelské procedury zapsané v produktu Java)
- Instance třídy uživatelské procedury MQExternalSend(pro uživatelskou proceduru odeslání, která není zapsána v produktu Java)
- Instance třídy řetězce MQSendExit.
- Instance třídy Řetězec

Seznam nemůže obsahovat jiný seznam.

Aplikace může použít posloupnost uživatelských procedur příjmu podobným způsobem.

Je-li použit řetězec, musí se skládat z jedné nebo více definic uživatelských procedur oddělených čárkami, z nichž každá může být názvem třídy Java nebo C programem ve formátu `library(function)`.

Aplikace pak přiřadí objekt List nebo String do pole MQEnvironment.channelSendExit před vytvořením objektu MQQueueManager .

Kontext informací předaných východům je výhradně v rámci domény východů. Je-li například ukončena uživatelská procedura Java a ukončení C, přítomnost procedury Java nemá žádný vliv na uživatelskou proceduru C.

Použití tříd uživatelských řetězců

Ve verzích starších než IBM WebSphere MQ 7.0byly k dispozici dvě třídy, které umožňují posloupnosti uživatelských procedur:

- Řetězec MQSendExit, který implementuje rozhraní MQSendExit
- Řetězec MQReceiveExit, který implementuje rozhraní MQReceiveExit

Použití těchto tříd zůstává v platnosti, ale dává přednost nové metodě. Použití třídy IBM MQ pro rozhraní Java znamená, že vaše aplikace stále má závislost na `com.ibm.mq.jar` Pokud se nová sada rozhraní v balíku `com.ibm.mq.exits` používá, není žádná závislost na `com.ibm.mq.jar`.

Chcete-li použít posloupnost uživatelských procedur odeslání, aplikace vytvořila seznam objektů, kde každý objekt byl jeden z následujících objektů:

- Instance třídy definované uživatelem, která implementuje rozhraní MQSendExit (pro odeslání uživatelské procedury zapsané v produktu Java)
- Instance třídy uživatelské procedury MQExternalSend(pro uživatelskou proceduru odeslání, která není zapsána v produktu Java)
- Instance třídy řetězce MQSendExit.

Aplikace vytvořila objekt řetězce MQSendExitpředáním tohoto seznamu objektů jako parametru v konstruktoru. Aplikace by poté před vytvořením objektu MQQueueManager přiřadil objekt řetězce MQSendExitk poli MQEnvironment.sendExit .

Komprese kanálu v produktu IBM MQ classes for Java

Komprimace dat, která proudí na kanálu, může zvýšit výkon kanálu a omezit provoz na síti. IBM MQ classes for Java použijte kompresní funkci zabudovaná do IBM MQ.

Pomocí funkce dodávané s produktem IBM MQmůžete komprimovat data, která se přenášejí na kanály zpráv a kanály MQI, a na obou typech kanálů můžete komprimovat data záhlaví a data zprávy nezávisle na ostatních. Standardně nejsou žádná data komprimována na kanálu. Úplný popis komprese kanálu včetně toho, jak je implementován v produktu IBM MQ, najdete v tématu Komprese dat (COMPMSG) a Komprese záhlaví (COMPHDR).

Aplikace IBM MQ classes for Java určuje techniky, které lze použít pro kompresi dat záhlaví nebo zprávy na připojení klienta vytvořením objektu java.util.Collection . Každá kompresní technika je objekt Integer v kolekci a pořadí, ve kterém aplikace přidá techniky komprese do kolekce, je pořadí, ve kterém jsou techniky komprese vyjednané se správcem front při spuštění připojení klienta. Aplikace pak může přiřadit kolekci do pole Seznam hdrComp, pro data záhlaví nebo pole seznamu msgComppro data zprávy, ve třídě MQEnvironment. Je-li aplikace připravena, může spustit připojení klienta vytvořením objektu MQQueueManager .

Následující fragmenty kódu popisují popsaný přístup. První fragment kódu ukazuje, jak implementovat kompresi dat záhlaví:

```
Collection headerComp = new Vector();
headerComp.add(new Integer(CMQXC.MQCOMPRESS_SYSTEM));
:
MQEnvironment.hdrCompList = headerComp;
:
MQQueueManager qMgr = new MQQueueManager(QM);
```

Druhý fragment kódu ukazuje, jak implementovat kompresi dat zprávy:

```
Collection msgComp = new Vector();
msgComp.add(new Integer(CMQXC.MQCOMPRESS_RLE));
msgComp.add(new Integer(CMQXC.MQCOMPRESS_ZLIBHIGH));
:
MQEnvironment.msgCompList = msgComp;
:
MQQueueManager qMgr = new MQQueueManager(QM);
```

Ve druhém příkladu jsou techniky komprese vyjednané v pořadí RLE, pak ZLIBHIGH, když se spustí připojení klienta. Zvolená technika komprese nemůže být změněna během doby životnosti objektu MQQueueManager .

Techniky komprese pro záhlaví a data zpráv, které jsou podporovány klientem i správcem front v připojení klienta, jsou předány do uživatelské procedury kanálu jako kolekce v polích hdrCompList a msgCompList objektu MQChannelDefinition . Skutečné techniky, které se aktuálně používají pro kompresi záhlaví

a dat zpráv v připojení klienta, jsou předány uživatelské proceduře kanálu v polích Komprese CurHdra komprese CurMsgobjektu MQChannelExit .

Je-li komprese použita na připojení klienta, data jsou komprimována před zpracováním a extrakcí kanálu odesílání kanálu po zpracování všech uživatelských procedur příjmu kanálu. Data předaná k odeslání a přijetí uživatelských procedur se proto nacházejí v komprimovaném stavu.

Další informace o určování technik komprese a o tom, které techniky komprese jsou k dispozici, naleznete v části Třída com.ibm.mq.MQEnvironment a Rozhraní com.ibm.mq.MQC.

Sdílení připojení TCP/IP v produktu IBM MQ classes for Java

Je možné vytvořit více instancí kanálu MQI, aby bylo možné sdílet jedno připojení TCP/IP.

V produktu IBM MQ classes for Javamůžete prostřednictvím proměnné MQEnvironment.sharingConversations řídit počet konverzací, které mohou sdílet jedno připojení TCP/IP.

Atribut SHARECNV je nejlepším přístupem k sdílení připojení. Proto je-li hodnota parametru SHARECNV větší než 0 použita v kombinaci s IBM MQ classes for Java , není zaručeno, že nový požadavek na připojení bude vždy sdílet již vytvořené spojení.

Fondy připojení v produktu IBM MQ classes for Java

Produkt IBM MQ classes for Java umožňuje sdružení volných připojení ve fondu pro opětovné použití.

Produkt IBM MQ classes for Java poskytuje dodatečnou podporu pro aplikace, které se zabývají více připojeními ke správcům front IBM MQ . Není-li připojení již vyžadováno, lze ji namísto zničení sloučit do fondu a později je znovu použít. To může poskytnout podstatné zvýšení výkonu pro aplikace a middleware, které se sériově připojují k libovolnému správci front.

IBM MQ poskytuje výchozí fond připojení. Aplikace mohou aktivovat nebo deaktivovat tento fond připojení registrací a deregistrováním tokenů prostřednictvím třídy MQEnvironment. Je-li fond aktivní, když produkt IBM MQ classes for Java konstruuje objekt MQQueueManager , prohledá tento výchozí fond a znovu použije jakékoli vhodné připojení. Pokud dojde k volání funkce MQQueueManager.disconnect (), je základní připojení vráceno do fondu.

Aplikace mohou alternativně vytvářet fond připojení MQSimpleConnectionManager pro konkrétní použití. Aplikace pak může buď tento fond zadat během konstrukce objektu MQQueueManager , nebo předat tento fond do prostředí MQEnvironment, aby jej bylo možné použít jako výchozí fond připojení.

Chcete-li zabránit připojení k použití příliš velkého množství prostředků, můžete omezit celkový počet připojení, která objekt správce MQSimpleConnectiondokáže zpracovat, a můžete omezit velikost fondu připojení. Nastavení limitů je užitečné, pokud existují konfliktní požadavky na připojení v rámci prostředí JVM.

Při výchozím nastavení metoda getMaxConnections () vrací hodnotu nula, což znamená, že neexistuje žádné omezení pro počet připojení, které objekt MQSimpleConnectionManager dokáže zpracovat. Limit můžete nastavit pomocí metody setMaxConnections (). Pokud jste nastavili limit a byl dosažen limit, požadavek na další připojení může způsobit, že bude vyvolána výjimka MQException s kódem příčiny MQRC_MAX_CONNS_LIMIT_REACHED.

Řízení výchozího fondu připojení v produktu IBM MQ classes for Java

Tento příklad ukazuje, jak používat výchozí fond připojení.

Prohlédněte si následující vzorovou aplikaci MQApp1:

```
import com.ibm.mq.*;
public class MQApp1
{
      public static void main(String[] args) throws MQException
      {
         for (int i=0; i<args.length; i++) {
            MQQueueManager qmgr=new MQQueueManager(args[i]);
            :
            : (do something with qmgr)
            :
            qmgr.disconnect();
```

```
        }
    }
}
```

Aplikace MQApp1 přijímá seznam lokálních správců front z příkazového řádku, připojuje se k jednotlivým správcům a provádí určitou operaci. Když však příkazový řádek zobrazuje vícekrát stejného správce front, je efektivnější připojit se pouze jednou a znovu použít toto připojení mnohokrát.

IBM MQ classes for Java poskytuje výchozí fond připojení, který můžete použít k provedení tohoto. Chcete-li fond povolit, použijte jeden z metod MQEnvironment.addConnectionPoolToken(). Chcete-li fond zakázat, použijte volbu MQEnvironment.removeConnectionPoolToken().

Následující ukázková aplikace, MQApp2, je funkčně identická s MQApp1, ale připojuje se k jednotlivým správcům front pouze jednou.

```
import com.ibm.mq.*;
public class MQApp2
{
    public static void main(String[] args) throws MQException
    {
       MQPoolToken token=MQEnvironment.addConnectionPoolToken();

       for (int i=0; i<args.length; i++) {
          MQQueueManager qmgr=new MQQueueManager(args[i]);
          :
          : (do something with qmgr)
          :
          qmgr.disconnect();
       }

       MQEnvironment.removeConnectionPoolToken(token);

    }
}
```

První smělý řádek aktivuje výchozí fond připojení registrací objektu MQPoolToken s rozhraním MQEnvironment.

Konstruktor MQQueueManager nyní prohledá tento fond pro příslušné připojení a vytvoří připojení ke správci front pouze v případě, že nemůže najít již existující připojení. Volání qmgr.disconnect() vrátí připojení do fondu pro pozdější použití. Tato volání rozhraní API jsou stejná jako ukázková aplikace MQApp1.

Druhý zvýrazněný řádek deaktivuje výchozí fond připojení, který zničí všechna připojení správce front uložená ve fondu. To je důležité, protože v opačném případě by aplikace byla ukončena s počtem aktivních připojení správců front ve fondu. Tato situace může způsobit chyby, které se objeví v protokolech správce front.

Pokud aplikace používá tabulku definic kanálů klienta (CCDT) k připojení ke správci front, konstruktor MQQueueManager nejprve prohledá tabulku pro vhodnou definici kanálu připojení klienta. Je-li nalezen, prohledá konstruktor výchozí fond připojení pro připojení, které lze použít pro kanál. Pokud konstruktor nemůže najít vhodné připojení ve fondu, prohledá tabulku definic kanálů klienta pro další vhodnou definici kanálu připojení klienta a bude pokračovat, jak je popsáno výše. Pokud konstruktor dokončí hledání v tabulce definic kanálů klienta a nepodaří se najít žádné vhodné připojení ve fondu, konstruktor spustí druhé prohledání tabulky. Během tohoto vyhledávání se konstruktor pokusí o vytvoření nového připojení pro každou vhodnou definici kanálu připojení klienta a použije první připojení, které se bude spravovat k vytvoření.

Výchozí fond připojení ukládá maximálně deset nevyužitých připojení a udržuje nepoužívaná připojení aktivní po dobu maximálně pěti minut. Aplikace může tuto změnu změnit (podrobnosti viz “Dodání jiného fondu připojení v produktu IBM MQ classes for Java” na stránce 358).

Místo použití produktu MQEnvironment k zadání objektu MQPoolTokenmůže aplikace sestavit vlastní:

```
   MQPoolToken token=new MQPoolToken();
   MQEnvironment.addConnectionPoolToken(token);
```

Některé aplikace nebo dodavatelé middlewaru poskytují podtřídy MQPoolToken za účelem předávání informací do vlastního fondu připojení. Mohou být konstruována a předána do addConnectionPoolToken() takovým způsobem, aby mohly být do fondu připojení předány další informace.

Výchozí fond připojení a více komponent v produktu IBM MQ classes for Java
Tento příklad ukazuje, jak přidat nebo odebrat položku MQPoolTokens ze statické sady registrovaných objektů MQPoolToken .

Produkt MQEnvironment uchovává statickou sadu registrovaných objektů MQPoolToken . Chcete-li přidat nebo odebrat položku MQPoolTokens z této sady, použijte následující metody:

- MQEnvironment.addConnectionPoolToken()
- MQEnvironment.removeConnectionPoolToken()

Aplikace se může skládat z mnoha komponent, které existují nezávisle a vykonávají práci pomocí správce front. V takové aplikaci by každá komponenta měla přidat objekt MQPoolToken do sady MQEnvironment po dobu jeho životnosti.

Například vzorová aplikace MQApp3 vytvoří deset podprocesů a spustí každou z nich. Každý podproces registruje svůj vlastní MQPoolToken, čeká po určitou dobu a potom se připojí ke správci front. Jakmile se podproces odpojí, odebere svůj vlastní MQPoolToken.

Výchozí fond připojení zůstane aktivní, je-li v sadě MQPoolTokensnastaven alespoň jeden token, takže zůstane aktivní po dobu trvání této aplikace. Aplikace nemusí udržovat hlavní objekt v celkovém řízení podprocesů.

```
import com.ibm.mq.*;
public class MQApp3
{
      public static void main(String[] args)
      {
         for (int i=0; i<10; i++) {
            MQApp3_Thread thread=new MQApp3_Thread(i*60000);
            thread.start();
         }
      }
}

class MQApp3_Thread extends Thread
{
      long time;

      public MQApp3_Thread(long time)
      {
         this.time=time;
      }

      public synchronized void run()
      {
         MQPoolToken token=MQEnvironment.addConnectionPoolToken();
         try {
            wait(time);
            MQQueueManager qmgr=new MQQueueManager("my.qmgr.1");
            :
            : (do something with qmgr)
            :
            qmgr.disconnect();
         }
         catch (MQException mqe) {System.err.println("Error occurred!");}
         catch (InterruptedException ie) {}

         MQEnvironment.removeConnectionPoolToken(token);
      }
}
```

Dodání jiného fondu připojení v produktu IBM MQ classes for Java
Tento příklad ukazuje, jak lze použít třídu **com.ibm.mq.MQSimpleConnectionManager** k zajištění jiného fondu připojení.

Tato třída poskytuje základní prostředky pro použití fondu připojení a aplikace mohou tuto třídu používat k přizpůsobení chování fondu.

Jakmile je vytvořena instance, lze v konstruktoru MQQueueManager zadat správce MQSimpleConnectionManager. Správce MQSimpleConnectionpoté spravuje připojení, které je základem konstruovaného objektu MQQueueManager. Pokud správce MQSimpleConnectionobsahuje vhodné připojení ve fondu, je toto připojení znovu použito a vráceno do správce MQSimpleConnectionza voláním funkce MQQueueManager.disconnect ().

Toto chování demonstruje následující fragment kódu:

```
MQSimpleConnectionManager myConnMan=new MQSimpleConnectionManager();
myConnMan.setActive(MQSimpleConnectionManager.MODE_ACTIVE);
MQQueueManager qmgr=new MQQueueManager("my.qmgr.1", myConnMan);
:
: (do something with qmgr)
:
qmgr.disconnect();

MQQueueManager qmgr2=new MQQueueManager("my.qmgr.1", myConnMan);
:
: (do something with qmgr2)
:
qmgr2.disconnect();
myConnMan.setActive(MQSimpleConnectionManager.MODE_INACTIVE);
```

Připojení, které je vytvořeno během prvního konstruktoru MQQueueManager , je uloženo v adresáři myConnMan po volání qmgr.disconnect(). Připojení je poté znovu použito při druhém volání konstruktoru MQQueueManager .

Druhý řádek umožňuje správce MQSimpleConnectionManager. Poslední řádek vypne správce MQSimpleConnectiona zničí veškerá spojení, která se nacházejí ve fondu. Správce MQSimpleConnectionje ve výchozím nastavení v MODE_AUTO, který je popsán později v této sekci.

Správce MQSimpleConnectionpřiděluje připojení k nejnověji využívaným základům a likviduje připojení na základě nejdéle nepoužitého základu. Při výchozím nastavení je připojení zničeno, pokud nebylo použito pět minut, nebo pokud ve fondu existuje více než deset nepoužívaných připojení. Tyto hodnoty můžete změnit voláním MQSimpleConnectionManager.setTimeout().

Můžete také nastavit produkt MQSimpleConnectionManager pro použití jako výchozí fond připojení, který má být použit v případě, že není v konstruktoru MQQueueManager zadán žádný správce Connection Manager.

Následující aplikace demonstruje toto:

```
import com.ibm.mq.*;
public class MQApp4
{
      public static void main(String []args)
      {
         MQSimpleConnectionManager myConnMan=new MQSimpleConnectionManager();
         myConnMan.setActive(MQSimpleConnectionManager.MODE_AUTO);
         myConnMan.setTimeout(3600000);
         myConnMan.setMaxConnections(75);
         myConnMan.setMaxUnusedConnections(50);
         MQEnvironment.setDefaultConnectionManager(myConnMan);
         MQApp3.main(args);
      }
}
```

Tučné čáry vytvářejí a konfigurují objekt MQSimpleConnectionManager. Konfigurace provede následující akce:

- Ukončí spojení, která se nepoužívají po dobu jedné hodiny.
- Omezuje počet připojení spravovaných produktem myConnMan až 75.
- Omezuje počet nevyužitých připojení ve fondu na 50.

- Nastavuje MODE_AUTO, což je výchozí nastavení. To znamená, že fond je aktivní pouze v případě, že se jedná o výchozího správce připojení a v sadě MQPoolTokens , které má v držení MQEnvironment, je alespoň jeden token.

Nový správce MQSimpleConnectionje poté nastaven jako výchozí správce připojení.

V posledním řádku volá aplikace funkci MQApp3.main(). Tím se spustí počet podprocesů, kde každý podproces používá IBM MQ nezávisle. Tyto podprocesy používají myConnMan, když navazují spojení.

Koordinace JTA/JDBC pomocí produktu IBM MQ classes for Java

Produkt IBM MQ classes for Java podporuje metodu MQQueueManager.begin (), která umožňuje produktu IBM MQ vystupovat jako koordinátor pro databázi, která poskytuje ovladač vyhovující standardu JDBC typu 2 nebo JDBC typu 4.

Tato podpora není k dispozici na všech platformách. Chcete-li zkontrolovat, které platformy podporují koordinaci JDBC , prohlédněte si téma Systémové požadavky pro IBM MQ.

Chcete-li použít podporu XA-JTA, musíte použít speciální knihovnu přepínačů JTA. Způsob použití této knihovny se liší v závislosti na tom, zda používáte produkt Windows nebo jednu z dalších platforem.

Konfigurace koordinace JTA/JDBC na systému Windows

Knihovna XA je dodávána jako knihovna DLL s názvem ve formátu `jdbcxxx.dll`.

Dodané `jdbcora12.dll` poskytuje kompatibilitu s Oracle 12C, pro instalaci serveru IBM MQ Windows .

V systémech Windows je knihovna XA dodávána jako úplná knihovna DLL. Název této knihovny DLL je `jdbcxxx.dll` , kde `xxx` označuje databázi, pro kterou byla knihovna přepínače kompilována. Tato knihovna se nachází v adresáři `java\lib\jdbc` nebo `java\lib64\jdbc` v instalaci produktu IBM MQ classes for Java . Musíte deklarovat knihovnu XA, která je také popsána jako zaváděcí soubor přepínače, do správce front. Použijte IBM MQ Explorer. V panelu vlastností správce front určete podrobnosti o souboru načtení přepínače v panelu vlastností správce front. Je třeba zadat pouze název knihovny. Příklad:

V případě databáze Db2 nastavte pole SwitchFile na: `dbcdb2`

Pro databázi Oracle nastavte pole SwitchFile na: `jdbcora`

Konfigurace koordinace JTA/JDBC na platformách jiných než Windows

Jsou dodány soubory objektů. Propojte příslušný soubor s použitím dodaného souboru Makefile a deklarujte jej pro správce front pomocí konfiguračního souboru.

Pro každý systém správy databází produkt IBM MQ poskytuje dva objektové soubory. Chcete-li vytvořit 32bitovou knihovnu přepínačů, musíte propojit jeden soubor objektu a propojit jiný objektový soubor a vytvořit 64bitovou knihovnu přepínačů. Pro Db2je název každého objektového souboru jdbcdb2.o a pro Oracleje název každého objektového souboru jdbcora.o.

Každý soubor objektu je třeba propojit s použitím příslušného souboru Makefile dodaného s produktem IBM MQ. Knihovna přepínače vyžaduje jiné knihovny, které mohou být uloženy v různých lokalitách na různých systémech. Knihovna přepínačů však nemůže k nalezení těchto knihoven použít proměnnou prostředí cesty ke knihovně, protože knihovna přepínačů je načtena správcem front, který je spuštěn v prostředí setuid. Dodaný soubor Makefile proto zaručuje, že knihovna přepínačů obsahuje úplné názvy cest těchto knihoven.

Chcete-li vytvořit knihovnu přepínačů, zadejte příkaz **make** s následujícím formátem. Chcete-li vytvořit 32bitovou knihovnu přepínačů, zadejte příkaz v adresáři /java/lib/jdbc instalace produktu IBM MQ . Chcete-li vytvořit 64bitovou knihovnu přepínačů, zadejte příkaz v adresáři /java/lib64/jdbc .

```
make DBMS
```

kde *DBMS* je systém správy databází, pro který vytváříte knihovnu přepínačů. Platné hodnoty jsou db2 pro Db2 a `oracle` pro Oracle.

Zde je uveden příklad příkazu **make** :

```
make db2
```

Všimněte si následujících bodů:

- Chcete-li spustit 32bitové aplikace, musíte pro každý systém správy databází, který používáte, vytvořit 32bitovou a 64bitovou knihovnu přepínačů. Chcete-li spustit 64bitové aplikace, musíte vytvořit pouze 64bitovou knihovnu přepínačů. Pro produkt Db2je název každé knihovny přepínače jdbcdb2 a v případě databáze Oracleje název každé knihovny přepínače jdbcora. Soubory Makefile zajišťují, aby 32bitové a 64bitové knihovny přepínačů byly uloženy v různých adresářích IBM MQ . 32bitovou knihovnu přepínačů je uložena v adresáři /java/lib/jdbc a 64bitová knihovna s přepínačem je uložena v adresáři / java/lib64/jdbc .
- Protože můžete produkt Oracle nainstalovat kdekoli na systému, soubory Makefile používají proměnnou prostředí ORACLE_HOME k vyhledání umístění, kde je nainstalována databáze Oracle .

Po vytvoření knihoven přepínače pro produkt Db2, Oraclenebo oba tyto knihovny je třeba je deklarovat správci front. Pokud konfigurační soubor správce front (qm.ini) již obsahuje stanzy XAResourceManager pro databáze Db2 nebo Oracle , musíte nahradit položku SwitchFile v každé sekci jedním z následujících způsobů:

Pro databázi Db2

```
SwitchFile=jdbcdb2
```

Pro databázi Oracle

```
SwitchFile=jdbcora
```

Neuvádějte plně kvalifikovaný název cesty buď 32bitové, nebo 64bitové knihovny přepínačů. Uveďte pouze název knihovny.

Pokud konfigurační soubor správce front již neobsahuje sekce XAResourceManager pro databáze Db2 nebo Oracle , nebo pokud chcete přidat další sekce XAResourceManager , prohlédněte si příručku Správa , kde získáte informace o tom, jak sestavit sekci XAResourceManager . Avšak každý záznam SwitchFile v nové stanze XAResourceManager musí být přesně tak, jak je popsáno dříve pro databázi Db2 nebo Oracle . Musíte také zahrnout položku `ThreadOfControl=PROCESS`.

Po aktualizaci konfiguračního souboru správce front a ujistit se, že byly nastaveny všechny příslušné proměnné prostředí databáze, můžete správce front restartovat.

Použití koordinace JTA/JDBC
Okódovali jste volání rozhraní API jako v dodaném příkladu.

Základní posloupnost volání rozhraní API pro uživatelskou aplikaci je:

```
  qMgr = new MQQueueManager("QM1")
  Connection con = qMgr.getJDBCConnection( xads );
  qMgr.begin()

 < Perform MQ and DB operations to be grouped in a unit of work >

  qMgr.commit() or qMgr.backout();
  con.close()
  qMgr.disconnect()
```

`xads` v rámci volání getJDBCConnection je implementace rozhraní XADataSource specifická pro databázi, která definuje podrobnosti o databázi, k níž se má připojit. Informace o tom, jak vytvořit příslušný objekt XADataSource pro předání do getJDBCConnection, najdete v dokumentaci k vaší databázi.

Dále je třeba aktualizovat cestu ke třídám s použitím vhodných souborů JAR specifických pro databázi pro provádění práce s produktem JDBC .

Pokud se musíte připojit k více databázím, musíte několikrát zavolat getJDBCConnection k provedení transakce přes několik různých připojení.

Existují dvě formy getJDBCConnection, které odrážejí dvě formy XADataSource.getXAConnection:

```
public java.sql.Connection getJDBCConnection(javax.sql.XADataSource xads)
  throws MQException, SQLException, Exception

public java.sql.Connection getJDBCConnection(XADataSource dataSource,
                                             String userid, String password)
  throws MQException, SQLException, Exception
```

Tyto metody deklarují výjimku ve svých klauzulích throws, aby se vyvarovali problémů s ověřovatelem prostředí JVM pro zákazníky, kteří nepoužívají funkce JTA. Skutečná vyvolaná výjimka je javax.transaction.xa.XAException , která vyžaduje, aby byl soubor jta.jar přidán do cesty ke třídě pro programy, které na ni dříve nevyžadovaly.

Chcete-li použít podporu JTA/JDBC , musíte do své aplikace zahrnout následující příkaz:

```
MQEnvironment.properties.put(CMQC.THREAD_AFFINITY_PROPERTY, new Boolean(true));
```

Známé problémy a omezení s koordinací JTA/JDBC
Některé z problémů a omezení podpory JTA/JDBC závisí na systému správy databází, který se používá, například testované ovladače JDBC se chovají odlišně, když je databáze ukončována, zatímco je spuštěna aplikace. Je-li připojení k databázi, kterou aplikace používá, přerušené, existují kroky, které může aplikace provést pro nové vytvoření nového připojení ke správci front a databázi, aby bylo možné tato nová připojení použít k provedení požadované transakční práce.

Vzhledem k tomu, že podpora JTA/JDBC volá ovladače JDBC , může mít implementace těchto ovladačů JDBC významný vliv na chování systému. Testované ovladače JDBC se budou chovat jinak, je-li ukončena činnost databáze, zatímco je spuštěna aplikace.

Důležité: Vždy se vyhýbejte náhlému ukončení činnosti databáze, zatímco existují aplikace, které drží otevřená připojení k databázi.

Poznámka: An IBM MQ classes for Java application must connect by using bindings mode to make IBM MQ act as a database coordinator.

Více oddílů XAResourceManager
Použití více než jednoho objektu stanza XAResourceManager v konfiguračním souboru správce front `qm.ini`není podporováno. Jakákoli sekce XAResourceManager jiná než první je ignorována.

Db2

Někdy Db2 vrátí chybu SQL0805N . Tento problém lze vyřešit pomocí následujícího příkazu příkazového procesoru:

```
DB2 bind @db2cli.lst blocking all grant public
```

Další informace naleznete v dokumentaci produktu Db2 .

Objekt stanza XAResourceManager musí být nakonfigurován pro použití ThreadOfControl=PROCESS. U Db2 8.1 a vyšší se to neshoduje s výchozím vláknem nastavení řízení pro Db2, takže toc=p musí být uvedeno v otevřeném řetězci XA. Příklad struktury XAResourceManager pro Db2 s koordinací JTA/JDBC je následující:

```
XAResourceManager:
     Name=jdbcdb2
     SwitchFile=jdbcdb2
     XAOpenString=uid=userid,db=dbalias,pwd=password,toc=p
     ThreadOfControl=PROCESS
```

Tím se nezabrání aplikacím produktu Java , které používají koordinaci JTA/JDBC , aby byly spuštěny s více podprocesy.

Oracle
Volání metody JDBC Connection.close() po funkci MQQueueManager.disconnect () generuje výjimku SQLException. Buď volejte Connection.close() před MQQueueManager.disconnect (), nebo vynechte volání na Connection.close().

Řešení problémů s databázovými připojeními

Když aplikace IBM MQ classes for Java používá podporu JTA/JDBC , kterou poskytuje produkt IBM MQ, obvykle provádí následující kroky:

1. Vytvoří nový objekt MQQueueManager , který bude představovat připojení ke správci front, který bude pracovat jako správce transakcí.
2. Vytvoří objekt XADataSource , který obsahuje podrobnosti o tom, jak se připojit k databázi, která bude uvedena v transakci.
3. Volá metodu MQQueueManager.getJDBCConnection(XADataSource) předáním XADataSource , který byl vytvořen dříve. To způsobí, že IBM MQ classes for Java vytvoří připojení k databázi.
4. Vyvolá metodu MQQueueManager.begin () ke spuštění transakce XA.
5. Provádí práci systému zpráv a databáze.
6. Jakmile je dokončena veškerá požadovaná práce, zavolá metodu MQQueueManager.commit (). Tím je dokončena transakce XA.
7. Je-li v tomto okamžiku požadována nová transakce XA, může aplikace opakovat kroky 4, 5 a 6.
8. Po dokončení aplikace by mělo zavřít připojení k databázi vytvořené v kroku 3 a poté volat metodu MQQueueManager.disconnect () a odpojit se od správce front.

Produkt IBM MQ classes for Java uchovává interní seznam všech databázových spojení, která byla vytvořena při volání aplikace MQQueueManager.getJDBCConnection(XADataSource). Pokud správce front potřebuje komunikovat s databází během zpracování transakce XA, provede se následující zpracování:

1. Správce front volá do produktu IBM MQ classes for Javaa předává podrobnosti o volání XA, které je třeba předat do databáze.
2. Produkt IBM MQ classes for Java potom vyhledá příslušné připojení v seznamu a poté použije toto připojení k toku volání XA do databáze.

Pokud dojde ke ztrátě připojení k databázi během tohoto zpracování, měla by aplikace:

1. Zavoláním metody MQQueueManager.backout () proveďte zálohu všech existujících prací, které byly provedeny v rámci transakce.
2. Zavřete databázové připojení. To by mělo způsobit, že IBM MQ classes for Java odstraní podrobnosti o přerušeném databázovém spojení ze svého vnitřního seznamu.
3. Odpojte se od správce front voláním metody MQQueueManager.disconnect ().
4. Vytvořte nové připojení ke správci front vytvořením nového objektu MQQueueManager .
5. Vytvořte nové připojení k databázi voláním metody MQQueueManager.getJDBCConnection(XADataSource).
6. Znovu proveďte transakční práci.

Díky tomu může aplikace znovu vytvořit nové připojení ke správci front a databázi a poté tato připojení používat k provedení požadované transakční práce.

Podpora TLS (Transport Layer Security) v produktu IBM MQ classes for Java

Klientské aplikace produktu IBM MQ classes for Java podporují šifrování TLS. Chcete-li používat šifrování TLS, je třeba poskytovatele prostředí JSSE.

Klientské aplikace IBM MQ classes for Java používající funkci TRANSPORT (CLIENT) podporují šifrování TLS. TLS poskytuje šifrování komunikace, ověření a integritu zpráv. Obvykle se používá k zabezpečení komunikace mezi dvěma rovnocennými partnery na Internetu nebo v rámci intranetu.

Produkt IBM MQ classes for Java používá k ošetření šifrování TLS produkt Java Secure Socket Extension (JSSE), a proto vyžaduje poskytovatele JSSE. Prostředí JVM JSE v1.4 má zabudovaný poskytovatel JSSE. Podrobnosti o tom, jak spravovat a ukládat certifikáty se mohou lišit od poskytovatele k poskytovateli. Informace o tomto tématu naleznete v dokumentaci k poskytovateli JSSE.

V tomto oddílu se předpokládá, že poskytovatel JSSE je správně nainstalován a nakonfigurován a že byly nainstalovány a zpřístupněny vhodné certifikáty pro poskytovatele JSSE.

Pokud klientská aplikace produktu IBM MQ classes for Java používá tabulku definic kanálů klienta (CCDT) k připojení ke správci front, přečtěte si téma “Použití tabulky definic kanálů klienta s IBM MQ classes for Java” na stránce 336.

Povolení zabezpečení TLS v produktu IBM MQ classes for Java
Chcete-li povolit TLS, zadejte sadu CipherSuite. Existují dva způsoby, jak určit sadu CipherSuite.

TLS je podporováno pouze pro připojení klienta. Chcete-li povolit TLS, je třeba určit CipherSuite , která má být použita při komunikaci se správcem front, a tato sada CipherSuite musí odpovídat sadě CipherSpec na cílovém kanálu. Kromě toho musí poskytovatel JSSE podporovat název CipherSuite . Hodnota CipherSuites se však liší od specifikace CipherSpecs a mají tedy různé názvy. Produkt “TLS CipherSpecs a CipherSuites v IBM MQ classes for Java” na stránce 368 obsahuje tabulku s mapováním CipherSpecs podporovaných produktem IBM MQ na ekvivalentní CipherSuites , jak je známo, na JSSE.

Chcete-li povolit TLS, zadejte sadu CipherSuite pomocí proměnné statického člena sady sslCipherpro prostředí MQEnvironment. Následující příklad se připojuje ke kanálu SVRCONN s názvem SECURE.SVRCONN.CHANNEL, která byla nastavena tak, aby vyžadovala zabezpečení TLS se sadou CipherSpec TLS_RSA_WITH_AES_128_CBC_SHA:

```
MQEnvironment.hostname       = "your_hostname";
MQEnvironment.channel        = "SECURE.SVRCONN.CHANNEL";
MQEnvironment.sslCipherSuite = "SSL_RSA_WITH_AES_128_CBC_SHA";
MQQueueManager qmgr = new MQQueueManager("your_Q_manager");
```

Ačkoli má kanál CipherSpec z TLS_RSA_WITH_AES_128_CBC_SHA, aplikace Java musí určovat sadu CipherSuite pro SSL_RSA_WITH_AES_128_CBC_SHA. Seznam mapování mezi CipherSpecs a CipherSuitesviz “TLS CipherSpecs a CipherSuites v IBM MQ classes for Java” na stránce 368 .

Aplikace může také určit sadu CipherSuite nastavením vlastnosti prostředí CMQC.SSL_CIPHER_SUITE_PROPERTY.

Případně můžete použít tabulku CCDT (Client Channel Definition Table). Další informace naleznete v tématu “Použití tabulky definic kanálů klienta s IBM MQ classes for Java” na stránce 336

Pokud vyžadujete připojení klienta pro použití sady CipherSuite , která je podporována poskytovatelem prostředí JSSE FIPS produktu IBM Java (IBMJSSEFIPS), může aplikace nastavit pole sslFipsRequired ve třídě MQEnvironment na `true`. Alternativně může aplikace nastavit vlastnost prostředí CMQC.SSL_FIPS_REQUIRED_PROPERTY. Výchozí hodnota je `false`, což znamená, že připojení klienta může použít libovolnou sadu CipherSuite , kterou podporuje produkt IBM MQ.

Pokud aplikace používá více než jedno připojení klienta, hodnota pole sslFips, která se použije, když aplikace vytvoří první připojení klienta, určuje hodnotu, která se použije, když aplikace vytvoří jakékoli následné připojení klienta. Proto, když aplikace vytvoří následné připojení klienta, hodnota požadovaného pole sslFipsje ignorována. Chcete-li použít jinou hodnotu pro pole Vyžadováno sslFips, musíte aplikaci restartovat.

Chcete-li se úspěšně připojit pomocí TLS, úložiště údajů o důvěryhodnosti JSSE musí být nastaveno s kořenovým certifikátem certifikační autority, ze kterého lze ověřit certifikát prezentovaný správcem front. Podobně, je-li vlastnost SSLClientAuth v kanálu SVRCONN nastavena na hodnotu MQSSL_CLIENT_AUTH_REQUIRED, musí úložiště klíčů JSSE obsahovat identifikační certifikát, kterému správce front důvěřuje.

Související informace
Federální standardy zpracování informací (FIPS) pro UNIX, Linux, and Windows

Použití rozlišujícího názvu správce front v produktu IBM MQ classes for Java
Správce front identifikuje sám sebe pomocí certifikátu TLS, který obsahuje rozlišující název (DN). Klientská aplikace IBM MQ classes for Java může tento rozlišující název použít k ujištění, že komunikuje se správným správcem front.

Vzorek DN se zadává pomocí proměnné názvu sslPeerproměnné MQEnvironment. Například nastavení:

```
MQEnvironment.sslPeerName = "CN=QMGR.*, OU=IBM, OU=WEBSPHERE";
```

umožňuje úspěšné připojení k úspěchu pouze v případě, že správce front předloží certifikát s názvem Common Name začínajícím QMGR., a alespoň dva názvy organizačních jednotek, přičemž první z nich musí být IBM a druhý WebSphere.

Je-li nastaven název sslPeer, připojení bude úspěšná pouze tehdy, je-li nastavena na platný vzor a správce front předloží odpovídající certifikát.

Aplikace může také určit rozlišující název správce front nastavením vlastnosti prostředí CMQC.SSL_PEER_NAME_PROPERTY. Další informace o rozlišujících názvech naleznete v tématu Rozlišovací jména.

Použití seznamů odvolaných certifikátů v produktu IBM MQ classes for Java
Určete seznamy odvolaných certifikátů, které mají být použity, prostřednictvím třídy java.security.cert.CertStore . IBM MQ classes for Java pak zkontroluje certifikáty proti uvedenému seznamu CRL.

Seznam zrušených certifikátů (CRL) je sada certifikátů, které byly odvolány, a to buď vydávající certifikační autoritou, nebo místní organizací. Seznamy CRL jsou obvykle hostovány na serverech LDAP. Při použití Java 2 v1.4může být server CRL uveden v době připojení a certifikát představený správcem front je před povolením připojení zkontrolován vůči seznamu odvolaných certifikátů. Další informace o seznamech zrušených certifikátů a produktu IBM MQnaleznete v části Práce se seznamy odvolaných certifikátů a seznamy odvolaných certifikátů a Přístup k seznamům CRL a ARL s produkty IBM MQ classes for Java a IBM MQ classes for JMS.

Poznámka: Chcete-li produkt CertStore úspěšně používat s názvem CRL hostovaným na serveru LDAP, ujistěte se, že vaše sada SDK (Software Development Kit) produktu Java je kompatibilní se seznamem CRL. Některé sady SDK vyžadují, aby seznam CRL odpovídal RFC 2587, které definuje schéma pro protokol LDAP v2. Většina serverů LDAP v3 používá místo toho RFC 2256.

Seznamy CRL, které se mají použít, jsou určeny prostřednictvím třídy java.security.cert.CertStore . Podrobné informace o tom, jak získat instance položky CertStore, naleznete v dokumentaci k této třídě. Chcete-li vytvořit úložiště CertStore na základě serveru LDAP, nejprve vytvořte instanci parametrů LDAPCertStoreinicializovanou s nastavením serveru a portu, které se mají použít. Příklad:

```
import java.security.cert.*;
CertStoreParameters csp = new LDAPCertStoreParameters("crl_server", 389);
```

Po vytvoření instance Parameters CertStorepoužijte statický konstruktor na CertStore k vytvoření CertStore typu LDAP:

```
CertStore cs = CertStore.getInstance("LDAP", csp);
```

Podporovány jsou také další typy CertStore (například Kolekce). Pro poskytnutí redundance jsou k dispozici pouze některé servery CRL, které mají identické informace CRL. Máte-li objekt CertStore pro každý z těchto serverů CRL, umístěte je do vhodné kolekce. Následující příklad ukazuje objekty CertStore umístěné v ArrayList:

```
import java.util.ArrayList;
Collection crls = new ArrayList();
crls.add(cs);
```

Tato kolekce může být nastavena do statické proměnné MQEnvironment sslCert, než se připojí k povolení kontroly CRL:

```
MQEnvironment.sslCertStores = crls;
```

Certifikát, který předkládá správce front při nastavení připojení, je ověřen následujícím způsobem:

1. První objekt CertStore v kolekci identifikovaný pomocí úložišť sslCertse používá k identifikaci serveru CRL.
2. Došlo k pokusu o kontaktování serveru CRL.
3. Je-li pokus úspěšný, prohledá se server na shodu certifikátu.
   a. Pokud má být certifikát odvolán, proces vyhledávání skončil a žádost o připojení selže s kódem příčiny MQRC_SSL_CERTIFICATE_REVOKED.
   b. Pokud certifikát nebyl nalezen, je proces vyhledávání znovu spuštěn a připojení je povoleno pokračovat.
4. Je-li pokus o kontaktování serveru neúspěšný, bude použit další objekt CertStore pro identifikaci serveru CRL a proces se opakuje z kroku 2.

   Pokud se jednalo o poslední položku CertStore v kolekci, nebo pokud kolekce neobsahuje žádné objekty CertStore , došlo k selhání procesu vyhledávání a požadavek na připojení se nezdařil s kódem příčiny MQRC_SSL_CERT_STORE_ERROR.

Objekt kolekce určuje pořadí, ve kterém jsou použita položka CertStores .

Kolekce CertStores může být také nastavena pomocí CMQC.SSL_CERT_STORE_PROPERTY. Tato vlastnost také umožňuje zadat jednu položku CertStore bez členství v kolekci.

Je-li úložiště sslCertnastaveno na hodnotu null, neprovádí se žádná kontrola CRL. Tato vlastnost je ignorována, není-li nastavena sada sslCipherSuite.

Opětovné dohadování tajného klíče v produktu IBM MQ classes for Java
Klientská aplikace IBM MQ classes for Java může řídit, kdy se tajný klíč, který se používá pro šifrování na klientském připojení, znovu vyjednává, z hlediska celkového počtu odeslaných a přijatých bajtů.

Aplikace to může provést jedním z následujících způsobů: Pokud aplikace používá více než jeden z těchto způsobů, použijí se obvyklá pravidla přednosti.

- Nastavením pole Počet sslResetve třídě MQEnvironment.
- Nastavením vlastnosti prostředí MQC.SSL_RESET_COUNT_PROPERTY v objektu Hashtable. Aplikace pak přiřadí tabulku hashtable do pole `properties` ve třídě MQEnvironment nebo předá hašovací tabulku objektu MQQueueManager do svého konstruktoru.

Hodnota pole sslResetCount nebo vlastnost prostředí MQC.SSL_RESET_COUNT_PROPERTY představuje celkový počet bajtů odeslaných a přijatých kódem klienta IBM MQ classes for Java před opětovným vyjednávením tajného klíče. Počet odeslaných bajtů je číslo před šifrováním a počet přijatých bajtů je číslo po dešifrování. Počet bajtů také obsahuje řídicí informace odeslané a přijaté klientem IBM MQ classes for Java .

Pokud je počet obnovení nulový, což je výchozí hodnota, tajný klíč není nikdy znovu vyjednáván. Počet obnovení je ignorován, pokud není zadán parametr CipherSuite .

Dodání přizpůsobené proměnné SSLSocketFactory v produktu IBM MQ classes for Java
Pokud používáte přizpůsobenou továrnu JSSE Socket Factory, nastavte vlastnost MQEnvironment.sslSocketFactory na přizpůsobenou továrnu objektů. Podrobnosti se liší mezi různými implementacemi JSSE.

Různé implementace JSSE mohou poskytovat různé funkce. Například specializovaná implementace JSSE může umožnit konfiguraci konkrétního modelu šifrovacího hardwaru. Kromě toho někteří poskytovatelé JSSE umožňují přizpůsobení úložiště klíčů a úložiště údajů o důvěryhodnosti podle programu nebo

umožňují změnu výběru certifikátu identity z úložiště klíčů. V prostředí JSSE jsou všechna tato přizpůsobení abstrahována ve třídě továrny javax.net.ssl.SSLSocketFactory.

Podrobné informace o tom, jak vytvořit přizpůsobenou implementaci SSLSocketFactory , najdete v dokumentaci k prostředí JSSE. Podrobnosti se liší od poskytovatele k poskytovateli, ale typická posloupnost kroků může být:

1. Vytvoření objektu SSLContext s použitím statické metody v SSLContext
2. Inicializujte tento SSLContext s odpovídajícími implementacemi KeyManager a TrustManager (vytvořenými z jejich vlastních továrních tříd).
3. Vytvoření objektu SSLSocketFactory z kontextu SSLContext

Máte-li objekt SSLSocketFactory , nastavte objekt MQEnvironment.sslSocketFactory na upravený objekt továrny. Příklad:

```
javax.net.ssl.SSLSocketFactory sf = sslContext.getSocketFactory();
MQEnvironment.sslSocketFactory = sf;
```

Produkt IBM MQ classes for Java používá tento parametr SSLSocketFactory pro připojení ke správci front produktu IBM MQ . Tuto vlastnost lze také nastavit pomocí parametru CMQC.SSL_SOCKET_FACTORY_PROPERTY. Je-li parametr sslSocketFactory nastaven na hodnotu null, použije se výchozí hodnota SSLSocketFactory prostředí JVM. Tato vlastnost je ignorována, není-li nastavena sada sslCipherSuite.

Použijete-li vlastní SSLSocketFactories, zvažte vliv sdílení připojení TCP/IP. Je-li sdílení připojení možné, není požadován nový soket pro SSLSocketFactory , a to i v případě, že by se soket vytvořil jiným způsobem v kontextu následného požadavku na připojení. Pokud má být například při následném připojení zobrazen jiný certifikát klienta, pak sdílení připojení nesmí být povoleno.

Provedení změn v úložišti klíčů nebo úložišti údajů o důvěryhodnosti JSSE v produktu IBM MQ classes for Java
Změníte-li úložiště klíčů nebo úložiště údajů o důvěryhodnosti prostředí JSSE, musíte provést určité akce, aby se změny projevily.

Pokud změníte obsah úložiště klíčů nebo úložiště údajů o důvěryhodnosti prostředí JSSE nebo změníte umístění souboru úložiště klíčů nebo souboru úložiště údajů o důvěryhodnosti, aplikace produktu IBM MQ classes for Java spuštěné v daném čase automaticky nevezmou tyto změny do paměti. Aby se změny projevily, musí být provedeny následující akce:

- Aplikace musí zavřít všechna svá připojení a zničit veškerá nepoužívaná připojení ve fondech připojení.
- Pokud poskytovatel JSSE ukládá informace z úložiště klíčů a úložiště údajů o důvěryhodnosti do mezipaměti, musí být tyto informace aktualizovány.

Po provedení těchto akcí mohou aplikace znovu vytvořit svá připojení.

V závislosti na návrhu aplikací a na funkci poskytované vaším poskytovatelem JSSE může být možné provést tyto akce bez zastavení a restartování aplikací. Avšak zastavení a restartování aplikací může být nejjednodušším řešením.

Ošetření chyb při použití TLS s IBM MQ classes for Java
Při připojování ke správci front pomocí TLS může produkt IBM MQ classes for Java vydat číslo kódu příčiny.

Ty jsou vysvětleny v následujícím seznamu:

MQRC_SSL_NOT_ALLOWED
Vlastnost sady sslCipherbyla nastavena, ale bylo použito připojení vazeb. TLS podporuje pouze připojení klienta.

CHYBA MQRC_JSSE_ERROR
Poskytovatel JSSE ohlásil chybu, kterou nelze zpracovat příkazem IBM MQ. Příčinou může být problém s konfigurací s podporou JSSE, nebo proto, že certifikát předložený správcem front nelze ověřit. Výjimka produkovaná JSSE může být načtena pomocí metody getCause() příkazu MQException.

CHYBA MQRC_SSL_INITIALIZATION_ERROR
Volání MQCONN nebo MQCONNX bylo vydáno s uvedenými volbami konfigurace TLS, ale během inicializace prostředí TLS došlo k chybě.

NESROVNALOST MQRC_SSL_PEER_NAME_
Vzorek DN zadaný ve vlastnosti názvu sslPeerneodpovídal rozlišujícímu názvu představenému správcem front.

CHYBA MQRC_SSL_PEER_NAME_ERROR
Vzorek DN zadaný ve vlastnosti sslPeerName nebyl platný.

MQRC_UNSUPPORTED_CIPHER_SUITE
Poskytovatel JSSE nerozeznal sadu CipherSuite uvedenou v sadě sslCipherSuite. Úplný seznam CipherSuites podporovaný poskytovatelem JSSE může být získán pomocí programu pomocí metody SSLSocketFactory.getSupportedCipherSuites(). Seznam CipherSuites , které lze použít ke komunikaci s produktem IBM MQ , lze nalézt v "TLS CipherSpecs a CipherSuites v IBM MQ classes for Java" na stránce 368.

MQRC_SSL_CERTIFICATE_ODVOLÁNO
Certifikát prezentovaný správcem front byl nalezen v seznamu odvolaných certifikátů, který je zadán ve vlastnosti sslCertStores. Aktualizujte správce front tak, aby používal důvěryhodné certifikáty.

CHYBA MQRC_SSL_CERT_STORE_ERROR
U žádného z dodaných CertStores nebylo možné hledat certifikát předložený správcem front. Metoda MQException.getCause() vrací chybu, která se vyskytla při hledání prvního pokusu CertStore . Je-li příčinná výjimka NoSuchElementException, ClassCastException, nebo NullPointerException, zkontrolujte, že kolekce uvedená ve vlastnosti sslCertStores obsahuje alespoň jeden platný objekt CertStore .

TLS CipherSpecs a CipherSuites v IBM MQ classes for Java
Schopnost aplikací produktu IBM MQ classes for Java vytvářet připojení ke správci front závisí na specifikaci CipherSpec na konci kanálu MQI a na straně klienta CipherSuite určenou na straně klienta.

V následující tabulce jsou uvedeny seznamy CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites.

Měli byste zkontrolovat téma Zamítnuto CipherSpecs a zjistit, zda některé ze specifikací CipherSpecsuvedené v následující tabulce byly zamítnuty produktem IBM MQ a, pokud ano, které aktualizaci CipherSpec byly zamítnuty.

Důležité: Seznam CipherSuites je seznam podporovaný produktem IBM Java Runtime Environment (JRE) dodávaným s produktem IBM MQ. Seznam CipherSuites , který je uveden, zahrnuje ty, které jsou podporovány prostředím JRE Oracle Java . Další informace o konfiguraci aplikace tak, aby používala prostředí Oracle Java JRE, viz "Konfigurace vaší aplikace pro použití mapování IBM Java nebo Oracle Java CipherSuite" na stránce 394.

Tabulka také uvádí protokol, který se používá pro komunikaci, a zda sada CipherSuite odpovídá standardu FIPS 140-2, či nikoli.

Pokud aplikace nebyla konfigurována k vynucení shody FIPS 140-2, lze použít specifikace Ciphersuites označené jako FIPS 140-2, ale pokud byl pro danou aplikaci nakonfigurován standard FIPS 140-2 (viz následující poznámky v konfiguraci), lze konfigurovat pouze ty CipherSuites , které jsou označeny jako kompatibilní s FIPS 140-2. Při pokusu o použití jiných sad CipherSuites se v chybě zobrazí chybová zpráva.

Poznámka: Každé prostředí JRE může mít více poskytovatelů zabezpečení šifrování, přičemž každý z nich může přispět k implementaci stejné sady CipherSuite. Nicméně ne všichni poskytovatelé zabezpečení jsou certifikováni FIPS 140-2. Pokud není pro aplikaci vynuceno dodržení standardu FIPS 140-2, je možné, že bude použita necertifikovaná implementace sady CipherSuite . Necertifikované implementace nemusí fungovat v souladu se standardem FIPS 140-2, a to ani v případě, že sada CipherSuite teoreticky splňuje minimální úroveň zabezpečení vyžadovanou standardem. Další informace o konfiguraci vynucení FIPS 140-2 v aplikacích produktu IBM MQ Java naleznete v následujících poznámkách.

Další informace o shodě s FIPS 140-2 a Suite-B pro CipherSpecs a CipherSuitesnajdete v tématu Uvedení CipherSpecs. Možná budete také potřebovat vědět o informacích, které se týkají Federální standardy zpracování informací.

Chcete-li použít úplnou sadu souborů CipherSuites a pracovat s certifikovaným standardem FIPS 140-2 anebo sadou Suite-B, je vyžadováno vhodné prostředí JRE. IBM Produkt Java 7 Service Refresh 4 Fix Pack 2 nebo vyšší úroveň prostředí IBM JRE poskytuje příslušnou podporu.

Poznámka: Chcete-li použít některé CipherSuites, je třeba v prostředí JRE nakonfigurovat 'neomezených' souborů zásad. Další podrobnosti o tom, jak jsou soubory zásad nastaveny v sadě SDK nebo JRE, viz téma *Soubory zásad sady SDK produktu IBM* v publikaci *Security Reference for IBM SDK, Java Technology Edition* pro verzi, kterou používáte.

Tabulka 59. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
ECDHE_ECDSA_3DES_EDE_CBC_SHA256	SSL_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA	TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA	TLSv1.2	yes

Tabulka 59. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
ECDHE_ECDSA_AES_128_CBC_SHA256	SSL_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	TLSv1.2	yes

Tabulka 59. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
ECDHE_ECDSA_AES_128_GCM_SHA256	SSL_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	TLSv1.2	yes

Tabulka 59. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
ECDHE_ECDSA_AES_256_CBC_SHA384	SSL_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	TLSv1.2	yes

Tabulka 59. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
ECDHE_ECDSA_AES_256_GCM_SHA384	SSL_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	TLSv1.2	yes

Tabulka 59. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
ECDHE_ECDSA_NULL_SHA256	SSL_ECDHE_ECDSA_WITHNULL_SHA	TLS_ECDHE_ECDSA_WITHNULL_SHA	TLSv1.2	ne

Tabulka 59. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
ECDHE_ECDSA_RC4_128_SHA256	SSL_ECDHE_ECDSA_WITH_RC4_128_SHA	TLS_ECDHE_ECDSA_WITH_RC4_128_SHA	TLSv1.2	ne

Tabulka 59. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
ECDHE_RSA_3DES_EDE_CBC_SHA256	SSL_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA	TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA	TLSv1.2	yes

Tabulka 59. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
ECDHE_RSA_AES_128_CBC_SHA256	SSL_ECDHE_RSA_WITH_AES_128_CBC_SHA256	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	TLSv1.2	yes

Tabulka 59. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	Ekvivalentní CipherSuite (IBM JRE)	Ekvivalentní CipherSuite (Oracle JRE)	Protokol	Kompatibilní s FIPS 140-2
ECDHE_RSA_AES_128_GCM_SHA256	SSL_ECDHE_RSA_WITH_AES_128_GCM_SHA256	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	TLSv1.2	yes

Tabulka 59. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
ECDHE_RSA_AES_256_CBC_SHA384	SSL_ECDHE_RSA_WITH_AES_256_CBC_SHA384	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	TLSv1.2	yes

Tabulka 59. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
ECDHE_RSA_AES_256_GCM_SHA384	SSL_ECDHE_RSA_WITH_AES_256_GCM_SHA384	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	TLSv1.2	yes

Tabulka 59. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
ECDHE_RSA_NULL_SHA256	SSL_ECDHE_RSA_WITH_NULL_SHA	TLS_ECDHE_RSA_WITH_NULL_SHA	TLSv1.2	ne

Tabulka 59. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
ECDHE_RSA_RC4_128_SHA256	SSL_ECDHE_RSA_WITH_RC4_128_SHA	TLS_ECDHE_RSA_WITH_RC4_128_SHA	TLSv1.2	ne

Tabulka 59. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
TLS_RSA_WITH_3DES_EDE_CBC_SHA "1" na stránce 393	SSL_RSA_WITH_3DES_EDE_CBC_SHA	TLS_RSA_WITH_3DES_EDE_CBC_SHA	TLSv1	yes

Tabulka 59. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
TLS_RSA_WITH_AES_128_CBC_SHA	SSL_RSA_WITH_AES_128_CBC_SHA	TLS_RSA_WITH_AES_128_CBC_SHA	TLSv1	yes

Tabulka 59. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
TLS_RSA_WITH_AES_128_CBC_SHA256	SSL_RSA_WITH_AES_128_CBC_SHA256	TLS_RSA_WITH_AES_128_CBC_SHA256	TLSv1.2	yes

Tabulka 59. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
TLS_RSA_WITH_AES_128_GCM_SHA256	SSL_RSA_WITH_AES_128_GCM_SHA256	TLS_RSA_WITH_AES_128_GCM_SHA256	TLSv1.2	yes

Tabulka 59. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
TLS_RSA_WITH_AES_256_CBC_SHA	SSL_RSA_WITH_AES_256_CBC_SHA	TLS_RSA_WITH_AES_256_CBC_SHA	TLSv1	yes

Tabulka 59. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
TLS_RSA_WITH_AES_256_CBC_SHA256	SSL_RSA_WITH_AES_256_CBC_SHA256	TLS_RSA_WITH_AES_256_CBC_SHA256	TLSv1.2	yes

Tabulka 59. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
TLS_RSA_WITH_AES_256_GCM_SHA384	SSL_RSA_WITH_AES_256_GCM_SHA384	TLS_RSA_WITH_AES_256_GCM_SHA384	TLSv1.2	yes

Tabulka 59. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
TLS_RSA_WITH_DES_CBC_SHA	SSL_RSA_WITH_DES_CBC_SHA	SSL_RSA_WITH_DES_CBC_SHA	TLSv1	ne

Tabulka 59. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	**Ekvivalentní CipherSuite (IBM JRE)**	**Ekvivalentní CipherSuite (Oracle JRE)**	**Protokol**	**Kompatibilní s FIPS 140-2**
TLS_RSA_WITH_NULL_SHA256	SSL_RSA_WITH_NULL_SHA256	TLS_RSA_WITH_NULL_SHA256	TLSv1.2	ne

Tabulka 59. CipherSpecs podporované produktem IBM MQ a jejich ekvivalenty CipherSuites (pokračování)

CipherSpec	Ekvivalentní CipherSuite (IBM JRE)	Ekvivalentní CipherSuite (Oracle JRE)	Protokol	Kompatibilní s FIPS 140-2
TLS_RSA_WITH_RC4_128_SHA256	SSL_RSA_WITH_RC4_128_SHA	SSL_RSA_WITH_RC4_128_SHA	TLSv1.2	ne

Notes:

1. Tato CipherSpec TLS_RSA_WITH_3DES_EDE_CBC_SHA byla zamítnuta. Nicméně lze ji přesto použít k přenosu až 32 GB dat před ukončením připojení s chybou AMQ9288. Chcete-li se této chybě vyhnout, je třeba při použití této CipherSpecbuď zabránit použití trojitého DES, nebo povolit resetování tajného klíče.

Konfigurace šifrovacích sad a shody s FIPS v aplikaci IBM MQ classes for Java

- Aplikace, která používá produkt IBM MQ classes for Java , může použít jednu ze dvou metod k nastavení CipherSuite pro připojení:
  - Nastavte pole Sada sslCipherve třídě MQEnvironment na název CipherSuite .
  - Nastavte vlastnost CMQC.SSL_CIPHER_SUITE_PROPERTY v tabulce hashtable předané do konstruktoru MQQueueManager do názvu CipherSuite .

- Aplikace, která používá produkt IBM MQ classes for Java , může použít některou ze dvou metod k vynucení shody FIPS 140-2:
  - Nastavte pole sslFipsRequired na hodnotu true ve třídě MQEnvironment.
  - Nastavte vlastnost CMQC.SSL_FIPS_REQUIRED_PROPERTYin hašovací tabulky vlastností předané konstruktoru MQQueueManager na hodnotu true.

Konfigurace vaší aplikace pro použití mapování IBM Java nebo Oracle Java CipherSuite

Můžete nakonfigurovat, zda vaše aplikace používá výchozí mapování produktu IBM Java CipherSuite na IBM MQ CipherSpec nebo mapování Oracle CipherSuite na IBM MQ CipherSpec . Z tohoto důvodu můžete použít TLS CipherSuites , zda vaše aplikace používá prostředí IBM JRE nebo Oracle JRE. Vlastnost systému Java `com.ibm.mq.cfg.useIBMCipherMappings` kontroluje, která mapování se používají. Vlastnost může mít jednu z následujících hodnot:

ano
Použijte IBM Java CipherSuite pro mapování IBM MQ CipherSpec .

Tato hodnota je výchozí hodnotou.

ne
Použijte mapování Oracle CipherSuite na IBM MQ CipherSpec .

Další informace o použití IBM MQ Java a šifer TLS naleznete v blogu MQdev MQ Java, TLS Šifry, Non-IBM JRE & APARs IT06775, IV66840, IT09423, IT10837.

Omezení interoperability

Určité CipherSuites mohou být kompatibilní s více než jedním IBM MQ CipherSpec, v závislosti na protokolu, který se používá. však je podporována pouze kombinace CipherSuite/CipherSpec , která používá verzi TLS uvedenou v tabulce 1. Pokus o použití nepodporované kombinace sad CipherSuites a CipherSpecs selže s příslušnou výjimkou. Instalace používající některou z těchto kombinací CipherSuite/CipherSpec by se měly přesunout do podporované kombinace.

Následující tabulka obsahuje sadu CipherSuites , na kterou se vztahuje toto omezení.

Tabulka 60. CipherSuites a jejich podporované a nepodporované CipherSpecs

CipherSuite	**Podporované TLS CipherSpec**	**Nepodporovaná SSL CipherSpec**
SSL_RSA_WITH_3DES_EDE_CBC_SHA	TLS_RSA_WITH_3DES_EDE_CBC_SHA "1" na stránce 394	TRIPLE_DES_SHA_US
SSL_RSA_WITH_DES_CBC_SHA	TLS_RSA_WITH_DES_CBC_SHA	DES_SHA_EXPORT
SSL_RSA_WITH_RC4_128_SHA	TLS_RSA_WITH_RC4_128_SHA256	RC4_SHA_US

Poznámka:

1. Tato CipherSpec TLS_RSA_WITH_3DES_EDE_CBC_SHA byla zamítnuta. Nicméně lze ji přesto použít k přenosu až 32 GB dat před ukončením připojení s chybou AMQ9288. Chcete-li se této chybě vyhnout, je třeba při použití této CipherSpecbuď zabránit použití trojitého DES, nebo povolit resetování tajného klíče.

Spuštění aplikací produktu IBM MQ classes for Java

Pokud napíšete aplikaci (třída, která obsahuje metodu main ()) pomocí klienta nebo režimu vazeb, spusťte program pomocí interpretačního programu Java .

Použijte příkaz:

```
java -Djava.library.path= library_path MyClass
```

kde *cesta_knihovny* je cesta ke knihovnám produktu IBM MQ classes for Java . Další informace viz “IBM MQ classes for Java knihovny” na stránce 318.

Související informace

Trasování aplikací produktu IBM MQ classes for Java

Trasování adaptéru prostředků IBM MQ

V 9.0.4 z/OS MQ Adv. VUE ***Připojitelnost klientů Java a JMS k dávkovým aplikacím spuštěným v systému z/OS***

JMSnebo IBM MQ classes for Javase aplikace v systému z/OS může připojit ke správci front v systému z/OS, který má atribut **ADVCAP**(ENABLED) pomocí připojení klienta.

Hodnota **ADVCAP** (ENABLED) se vztahuje pouze na správce front z/OS , který je licencován jako IBM MQ Advanced for z/OS, Value Unit Edition (viz IBM MQ product identifiers and export information), běží s **QMGRPROD** nastaveným na ADVANCEDVUE.

Další informace o příkazu **ADVCAP** a START QMGR naleznete v části DISPLAY QMGR , kde získáte další informace o produktu **QMGRPROD**.

Všimněte si, že dávka je jediným podporovaným prostředím; neexistuje žádná podpora pro JMS pro CICS nebo JMS pro IMS.

Aplikace IBM MQ classes for JMSnebo IBM MQ classes for Javase v produktu z/OS nemůže připojit pomocí připojení v režimu klienta ke správci front, který není spuštěn v produktu z/OS, ani ke správci front, který nemá volbu **ADVCAP** (ENABLED) .

Pokusí-li se aplikace JMS o připojení pomocí režimu klienta a aplikace není povolena, vydá se zpráva výjimky JMSFMQ0005 .

Pokud se aplikace IBM MQ classes for Java na z/OS pokusí o připojení pomocí režimu klienta a není to povoleno, vrátí se MQRC_ENVIRONMENT_ERROR .

Podpora produktu Advanced Message Security (AMS)

V 9.0.5

Z klientských aplikací IBM MQ 9.0.5, IBM MQ classes for JMS nebo IBM MQ classes for Java lze použít AMS při připojování ke správcům front IBM MQ Advanced for z/OS, Value Unit Edition na vzdálených systémech z/OS .

Nový typ úložiště klíčů, `jceracfks`, je podporován pouze v produktu `keystore.conf` v systému z/OS , kde:

- Předpona názvu vlastnosti je `jceracfks` a tato předpona názvu nerozlišuje velká a malá písmena.
- Úložiště klíčů je svazek klíčů RACF.
- Hesla se nepožadují a budou ignorována. Důvodem je to, že soubory keyrings RACF nepoužívají hesla.
- Určíte-li poskytovatele, poskytovatel musí být `IBMJCE`.

Když používáte produkt `jceracfks` s produktem AMS, musí být úložiště klíčů ve tvaru: `safkeyring://user/keyring`, kde:

- `safkeyring` je literál a tento název nerozlišuje velká a malá písmena.
- *user* je ID uživatele RACF, který vlastní svazek klíčů
- *keyring* je název svazku klíčů RACF a název svazku klíčů je citlivý na velikost písmen

V následujícím příkladu je použit standardní svazek klíčů AMS pro uživatele JOHNDOE:

```
jceracfks.keystore=safkeyring://JOHNDOE/drq.ams.keyring
```

Chování prostředí IBM MQ classes for Java závislé na prostředí

Produkt IBM MQ classes for Java vám umožňuje vytvářet aplikace, které mohou být spuštěny proti různým verzím produktu IBM MQ. Tato kolekce témat popisuje chování tříd produktu Java závislých na těchto různých verzích.

IBM MQ classes for Java poskytuje jádro tříd, které poskytují konzistentní funkci a chování ve všech prostředích. Funkce mimo toto jádro závisejí na schopnosti správce front, ke kterému je aplikace připojena.

Není-li zde uvedeno jinak, vykazuje se chování, jak je popsáno v tématu Odkaz na aplikaci MQI , který odpovídá správci front.

Hlavní třídy v produktu IBM MQ classes for Java

Produkt IBM MQ classes for Java obsahuje základní sadu tříd, kterou lze použít ve všech prostředích.

Následující sada tříd je považována za základní třídy a lze ji použít ve všech prostředích pouze s menšími variantami uvedenými v seznamu "Omezení a variace pro třídy jádra produktu IBM MQ classes for Java" na stránce 397.

- Prostředí MQEnvironment
- Výjimka MQException
- Volby MQGetMessage

  S výjimkou:

  - MatchOptions
  - GroupStatus
  - SegmentStatus
  - Segmentace
- MQManagedObject

  S výjimkou:

  - dotázat ()
  - nastavit ()
- Zpráva MQMessage

  S výjimkou:

  - groupId
  - messageFlags
  - messageSequenceČíslo
  - posunutí
  - originalLength
- MQPoolServices
- Událost MQPoolServices
- MQPoolServicesEventListener
- MQPoolToken
- Volby MQPutMessage

  S výjimkou:

  - KnownDestCount
  - UnknownDestCount
  - InvalidDestCount
  - recordFields

- Proces MQProcess
- MQQUEUE
- MQQueueManager

  S výjimkou:

  – začátek ()
  – accessDistributionList ()
- Správce MQSimpleConnectionManager
- MQTopic
- MQC

Poznámka:

1. Některé konstanty nejsou zahrnuty do jádra (podrobnosti viz “Omezení a variace pro třídy jádra produktu IBM MQ classes for Java” na stránce 397); nepoužívat je v kompletních přenosných programech.
2. Některé platformy nepodporují všechny režimy připojení. Na těchto platformách můžete používat pouze základní třídy a volby, které souvisejí s podporovanými režimy.

Omezení a variace pro třídy jádra produktu IBM MQ classes for Java

Hlavní třídy se obecně chovají konzistentně ve všech prostředích, a to i v případě, že ekvivalentní volání MQI normálně mají rozdíly prostředí. The behavior is as if a Windows, UNIX or Linux IBM MQ queue manager is used, except for the following minor restrictions and variations.

*Omezení pro hodnoty MQGMO_ * v produktu IBM MQ classes for Java*
Určité hodnoty MQGMO_ * nejsou podporovány všemi správci front.

Při použití následujících hodnot MQGMO_ * může dojít k vyvolání výjimky MQException z objektu MQQueue.get():

MQGMO_SYNCPOINT_IF_PERSISTENT
MQGMO_MARK_ SKIP_BACKOUT
MQGMO_BROWSE_MSG_UNDER_CURSOR
MQGMOVÝ_ZÁMEK
MQGMO_ODEMKNOUT
MQGMO_LOGICAL_ORDER
ZPRÁVA MQGMO_COMPLETE_MESSAGE
MQGMO_ALL_MSGS_AVAILABLE
DOSTUPNÉ MQGMO_ALL_SEGMENTS_AVAILABLE
MQGMO_UNMARKED_BROWSE_MSG,
POPISOVAČ MQGMO_MARK_ BROWSE_HANDLE
MQGMO_MARKER_BROWSE_CO_OP
POPISOVAČ MQGMO_UNMARK_ BROWSE_HANDLE
MQGMO_UNMARK_ BROWSE_CO_OP

Kromě toho není funkce MQGMO_SET_SIGNAL v případě použití z produktu Javapodporována.

*Omezení pro hodnoty MQPMRF_ * v produktu IBM MQ classes for Java*
Ty se používají pouze při vkládání zpráv do distribučního seznamu a jsou podporovány pouze správci front podporujícím distribuční seznamy. Například, správci front produktu z/OS nepodporují distribuční seznamy.

*Omezení pro hodnoty MQPMO_ * v produktu IBM MQ classes for Java*
Některé hodnoty MQPMO_ * nejsou podporovány všemi správci front

Použití následujících hodnot MQPMO_ * může vést k vyvolání výjimky MQException z objektu MQQueue.put() nebo objektu MQQueueManager.put ():

MQPMO_LOGICAL_ORDER
MQPMO_NOVÉ_ KOREL_ID
MQPMO_NOVÉ_ID_ZPRÁVY
MQPMOD_RESOLVE_LOKÁLNÍ_Q

*Omezení a variace pro hodnoty MQCNO_ * v produktu IBM MQ classes for Java*
Určité hodnoty MQCNO_* nejsou podporovány.

- Automatické opětovné připojení klienta není podporováno produktem IBM MQ classes for Java. Bez ohledu na hodnotu MQCNO_RECONNECT_* , kterou jste nastavili, se bude nadále chovat jako, jako byste nastavili MQCNO_RECONNECT_DISABLED.
- Produkt MQCNO_FASTPATH je ignorován ve správcích front, které nepodporují produkt MQCNO_FASTPATH. Je také ignorován klientskými připojeními.

*Omezení pro hodnoty MQRO_ * v produktu IBM MQ classes for Java*
Mohou být nastaveny následující volby sestavy.

MQRO_EXCEPTION_WITH_FULL_DATA
MQRO_EXPIRATION_WITH_FULL_DATA
MQRO_COA_WITH_FULL_DATA
MQRO_COD_WITH_FULL_DATA
MQRO_DISCARD_MSG
MQRO_PASS_DISCARD_AND_EXPIRY

Další informace viz Sestava.

Různé rozdíly mezi IBM MQ classes for Java na z/OS a jiných platformách
IBM MQ for z/OS se chová jinak než IBM MQ na jiných platformách v některých oblastech.

BackoutCount
Správce front produktu z/OS vrátí maximální hodnotu BackoutCount 255, a to i v případě, že zpráva byla zálohována více než 255 krát.

Výchozí předpona dynamické fronty
Při připojení ke správci front produktu z/OS s použitím připojení vazeb je výchozí předpona dynamické fronty CSQ. *. V opačném případě bude použita výchozí předpona dynamické fronty AMQ. *.

Konstruktor MQQueueManager
Připojení klienta není v produktu z/OSpodporováno. Pokus o připojení k volbám klienta vede k výjimce MQException s MQCC_FAILED a MQRC_ENVIRONMENT_ERROR.

Konstruktor MQQueueManager může také selhat s chybou MQRC_CHAR_CONVERSION_ERROR (pokud se nezdaří inicializovat převod mezi kódovými stránkami IBM-1047 a ISO8859-1) nebo MQRC_UCS2_CONVERSION_ERROR (pokud selže inicializace převodu mezi kódovou stránkou správce front a Unicode). Pokud vaše aplikace selže s jedním z těchto kódů příčiny, ujistěte se, že je nainstalována komponenta Jazykové prostředí národního prostředí a ujistěte se, že jsou k dispozici správné převodní tabulky.

Převodní tabulky pro kódování Unicode jsou nainstalovány jako součást volitelné funkce z/OS C/C++. Další informace o povolení převodů UCS-2 naleznete v příručce *z/OS C/C++ Programming Guide*, SC09-4765.

Funkce mimo hlavní třídy produktu IBM MQ classes for Java

IBM MQ classes for Java obsahuje určité funkce, které jsou speciálně navržené pro použití rozšíření rozhraní API, která nejsou podporována všemi správci front. Tato kolekce témat popisuje, jak se chovají při používání správce front, který je nepodporuje.

Variace ve volbě konstruktoru MQQueueManager
Některé z konstruktorů MQQueueManager obsahují volitelný celočíselný argument. Některé hodnoty tohoto argumentu nejsou akceptovány na všech platformách.

Pokud konstruktor MQQueueManager obsahuje volitelný celočíselný argument, mapuje se do pole voleb MQCNO rozhraní MQI a používá se k přepínání mezi normálním a rychlým připojením cesty. Tato přídavná forma konstruktoru je přijata ve všech prostředích, pokud jsou jediné použité volby MQCNO_STANDARD_BINDING nebo MQCNO_FASTPATH_BINDING. Všechny ostatní volby způsobí selhání konstruktoru při chybě MQRC_OPTIONS_ERROR. Volba rychlé cesty CMQC.MQCNO_FASTPATH_BINDING je dodržován pouze s vazbami připojení ke správci front, který ji podporuje. V jiných prostředích je ignorována.

Omezení pro metodu .begin () produktu MQQueueManager
Tuto metodu lze použít pouze pro správce front produktu IBM MQ v systémech UNIX, Linuxnebo Windows v režimu vazeb. Jinak dojde k selhání funkce MQRC_ENVIRONMENT_ERROR.

Další informace viz část "Koordinace JTA/JDBC pomocí produktu IBM MQ classes for Java" na stránce 360.

Variace v polích Volby MQGetMessage
Někteří správci front nepodporují strukturu MQGMO verze 2, takže je třeba nastavit některá pole na jejich výchozí hodnoty.

Používáte-li správce front, který nepodporuje strukturu MQGMO verze 2, ponechejte následující pole nastavovaná na výchozí hodnoty:

GroupStatus
SegmentStatus
Segmentace

Také pole MatchOptions podporuje pouze MQMO_MATCH_MSG_ID a MQMO_MATCH_CORREL_ID. Pokud do těchto polí zadáte nepodporované hodnoty, následující MQDestination.get() selže s chybou MQRC_GMO_ERROR. Pokud správce front nepodporuje strukturu MQGMO verze 2, tato pole se neaktualizují po úspěšném provedení operace MQDestination.get().

Omezení v distribučních seznamech v produktu IBM MQ classes for Java
Ne všichni správci front umožňují otevřít objekt MQDistributionList.

K vytváření distribučních seznamů se používají následující třídy:

MQDistributionList
Položka MQDistributionList
MQMessageTracker

Můžete vytvořit a naplnit položky MQDistributionLists a MQDistributionListv libovolném prostředí, ale ne všechny správce front umožňují otevřít MQDistributionList. Správci front produktu z/OS zejména nepodporují distribuční seznamy. Pokus o otevření objektu MQDistributionList při použití takového správce front vede k výjimce MQRC_OD_ERROR.

Variace v polích Volby MQPutMessage
Pokud správce front nepodporuje distribuční seznamy, budou s některými poli MQPMO zacházeno jinak.

Čtyři pole v objektu MQPMO jsou vykreslena jako následující členské proměnné ve třídě Volby MQPutMessage:

KnownDestCount
UnknownDestCount
InvalidDestCount
recordFields

Tato pole jsou primárně určena pro použití s rozdělovníky. Avšak správce front, který podporuje distribuční seznamy, vyplní také pole DestCount po provedení MQPUT do jediné fronty. Například, pokud je fronta vyřešena na lokální frontu, knownDestPočet je nastaven na 1 a ostatní dvě pole počtu jsou nastavena na 0.

Pokud správce front nepodporuje distribuční seznamy, jsou tyto hodnoty simulovány následujícím způsobem:

- Je-li funkce put () úspěšná, unknownDestCount je nastaven na 1 a ostatní jsou nastaveny na 0.
- Pokud funkce put () selže, invalidDestPočet je nastaven na hodnotu 1 a ostatní jsou nastaveny na 0.

Proměnná recordFields se používá s rozdělovníky. Hodnota může být kdykoli zapsána do recordFields , bez ohledu na prostředí. Pokud je objekt Volby MQPutMessagepoužit v následném objektu MQDestination.put() nebo MQQueueManager.put () a nikoli MQDistributionList.put (), je tento parametr ignorován.

Omezení v polích MQMD s IBM MQ classes for Java
Při použití správce front, který nepodporuje segmentaci, by se při použití správce front, která nepodporuje segmentaci, měla při výchozím nastavení ponechat určitá pole MQMD se segmentací zpráv.

Následující pole MQMD jsou z velké části znepokojují segmentací zpráv:

GroupId
MsgSeqNumber
Offset
MsgFlags
OriginalLength

Pokud aplikace nastaví jakékoli z těchto polí MQMD na jiné hodnoty než jejich výchozí hodnoty a pak funkci put () nebo get () ve správci front, který tyto hodnoty nepodporuje, vrátí funkce put () nebo get () výjimku MQException s MQRC_MD_ERROR. Úspěšná funkce put () nebo get () s takovým správcem front vždy ponechá pole MQMD nastavenou na jejich výchozí hodnoty. Neposílejte seskupenou nebo segmentovanou zprávu do aplikace produktu Java , která je spuštěna proti správci front, který nepodporuje seskupování zpráv a segmentaci.

Pokusí-li se aplikace Java získat () zprávu ze správce front, který tato pole nepodporuje, a fyzická zpráva, která má být načtena, je součástí skupiny segmentovaných zpráv (to znamená, že má jiné než výchozí hodnoty pro pole MQMD), je načtena bez chyb. Nicméně pole MQMD ve zprávě MQMessage nejsou aktualizována, vlastnost formátu MQMessage je nastavena na hodnotu MQFMT_MD_EXTENSION a před daty ve struktuře MQMDE, která obsahuje hodnoty pro nová pole, bude použita předpona MQMMT_MD_EXTENSION.

Omezení pro produkt IBM MQ classes for Java v rámci komponenty CICS Transaction Server

V prostředí CICS Transaction Server pro prostředí z/OS je povolen pouze hlavní (první) podproces za účelem vyvolání volání CICS nebo IBM MQ .

Všimněte si, že třídy IBM MQ JMS nejsou podporovány pro použití v aplikaci CICS Java .

Z tohoto důvodu není možné sdílet objekty MQQueueManager nebo MQQueue mezi podprocesy v tomto prostředí nebo vytvořit nový objekt MQQueueManager v podřízeném podprocesu.

z/OS "Různé rozdíly mezi IBM MQ classes for Java na z/OS a jiných platformách" na stránce 398 identifikuje některá omezení a varianty, které se vztahují na produkt IBM MQ classes for Java při spuštění se správcem front z/OS . Kromě toho nejsou při spuštění v produktu CICSpodporovány metody řízení transakcí ve správci front MQQueueManager . Místo zadání příkazu MQQueueManager.commit () nebo MQQueueManager.backout () používají aplikace metody synchronizace úloh JCICS , Task.commit() a Task.rollback(). Třída Úloha je dodána produktem JCICS v balíku com.ibm.cics.server .

Použití adaptéru prostředků produktu IBM MQ

Adaptér prostředků umožňuje aplikacím běžícím na aplikačním serveru přistupovat k prostředkům produktu IBM MQ . Podporuje příchozí a odchozí komunikaci.

Co obsahuje adaptér prostředku

Produkt Java Platform, Enterprise Edition Connector Architecture (JCA) poskytuje standardní způsob připojení aplikací, které jsou spuštěny v prostředí produktu Java EE , k podnikovému informačnímu

systému (EIS), jako např. IBM MQ nebo Db2. Adaptér prostředků IBM MQ implementuje rozhraní JCA 1.7 a obsahuje IBM MQ classes for JMS. Umožňuje aplikacím produktu JMS a objektům typu message-driven bean (MDB) spuštěným na aplikačním serveru přistupovat k prostředkům správce front produktu IBM MQ . Adaptér prostředků podporuje doménu typu point-to-point i doménu publikování/odběru.

Adaptér prostředků IBM MQ podporuje dva typy komunikace mezi aplikací a správcem front:

Odchozí komunikace
Aplikace spustí připojení ke správci front a poté odesílá zprávy produktu JMS do míst určení JMS a přijímá zprávy JMS z cílů JMS synchronním způsobem.

Příchozí komunikace
Zpráva JMS , která dorazí do cíle JMS , je doručena do objektu MDB, který asynchronně zpracuje zprávu.

Adaptér prostředku také obsahuje IBM MQ classes for Java. Třídy jsou automaticky dostupné pro aplikace spuštěné na aplikačním serveru, do kterého byl implementován adaptér prostředků, a povolit aplikacím spuštěným v daném aplikačním serveru používat rozhraní API produktu IBM MQ classes for Java při přístupu k prostředkům správce front IBM MQ .

Použití IBM MQ classes for Java v prostředí Java EE je podporováno s omezeními. Informace o těchto omezeních viz "Spuštění aplikací IBM MQ classes for Java v rámci produktu Java EE" na stránce 310.

Kterou verzi adaptéru prostředku použít

Verze produktu Java Platform, Enterprise Edition (Java EE) aplikačního serveru, kterou používáte, určuje verzi adaptéru prostředků, kterou musíte použít:

Java EE 7
Adaptér prostředků IBM MQ 8.0 a IBM MQ 9.0 podporuje JCA v1.7 a poskytuje podporu JMS 2.0 . Tento adaptér prostředků musí být implementován v rámci aplikačního serveru Java EE 7 a novější (viz "Příkaz podpory adaptéru prostředků produktu IBM MQ" na stránce 402).

Adaptér prostředků produktu IBM MQ 8.0 nebo novější můžete nainstalovat na kterýkoli aplikační server, který je certifikován jako vyhovující specifikaci Java Platform, Enterprise Edition 7 . Pomocí adaptéru prostředků produktu IBM MQ 8.0 nebo novější se aplikace může připojit ke správci front produktu IBM WebSphere MQ 7.0 nebo novější buď pomocí přenosu BINDINGS, nebo CLIENT, nebo pomocí správce front produktu IBM WebSphere MQ 6.0 pouze pomocí přenosu CLIENT.

Důležité: Adaptér prostředků produktu IBM MQ 8.0 nebo novější lze implementovat pouze na aplikační server, který podporuje produkt JMS 2.0.

Java EE 5 a Java EE 6
Adaptér prostředků IBM WebSphere MQ 7.5 podporuje produkt Java EE Connector Architecture (JCA) v1.5 a poskytuje podporu JMS 1.1 . Chcete-li zajistit úplnou integraci s produktem WebSphere Application Server Liberty, adaptér prostředků produktu IBM WebSphere MQ 7.5 se aktualizuje na APAR IC92914 z produktu IBM WebSphere MQ 7.5.0 Fix Pack 2. Tento adaptér prostředků zachovává plnou kompatibilitu s ostatními aplikačními servery Java EE 5 a novějších (viz WebSphere MQ resource adapter 7.1 and later statement of support).

Použití adaptéru prostředků s produktem WebSphere Application Server traditional

V 9.0.0 Adaptér prostředků produktu IBM MQ 9.0 je předinstalován v rámci produktu WebSphere Application Server traditional 9.0. Proto není třeba instalovat nový adaptér prostředků.

Poznámka: Adaptér prostředků produktu IBM MQ 9.0 se může v režimu transportu CLIENT nebo BINDINGS připojit k libovolnému správci front v produktu IBM MQ .

Použití adaptéru prostředků s produktem WebSphere Application Server Liberty

Chcete-li se připojit k produktu IBM MQ z produktu WebSphere Application Server Liberty, musíte použít adaptér prostředků produktu IBM MQ . Vzhledem k tomu, že produkt Liberty neobsahuje adaptér

prostředků IBM MQ , je třeba jej získat odděleně od Fix Central. Verze adaptéru prostředků, kterou používáte, závisí na verzi produktu Java EE aplikačního serveru.

Další informace o tom, jak stahovat a instalovat adaptér prostředků, viz “Instalace adaptéru prostředků v produktu Liberty” na stránce 409.

Související pojmy

“Konfigurace adaptéru prostředků pro příchozí komunikaci” na stránce 415
Chcete-li nakonfigurovat příchozí komunikaci, definujte vlastnosti jednoho nebo více objektů ActivationSpec .

“Konfigurace adaptéru prostředků pro odchozí komunikaci” na stránce 431
Chcete-li nakonfigurovat odchozí komunikaci, definujte vlastnosti objektu ConnectionFactory a spravovaného cílového objektu.

“použitíIBM MQ classes for JMS” na stránce 69
IBM MQ classes for Java Message Service (IBM MQ classes for JMS) je poskytovatel JMS , který je dodáván s IBM MQ. Stejně jako implementace rozhraní definovaných v balíku javax.jms poskytuje produkt IBM MQ classes for JMS dvě sady rozšíření rozhraní API produktu JMS .

“použitíIBM MQ classes for Java” na stránce 308
Použijte IBM MQ v prostředí Java . IBM MQ classes for Java umožňuje aplikaci Java připojit se k IBM MQ jako klient IBM MQ nebo se připojit přímo ke správci front IBM MQ .

Související informace

Konfigurace aplikačního serveru pro použití nejnovější úrovně údržby adaptéru prostředků

Určování problémů pro adaptér prostředků produktu IBM MQ

Související informace pro produkt WebSphere Application Server verze 8.5.5

Údržba adaptéru prostředků produktu IBM MQ

Implementace aplikací produktu JMS do profilu Liberty pro použití poskytovatele systému zpráv produktu IBM MQ

Příkaz podpory adaptéru prostředků produktu IBM MQ

Adaptér prostředků, který je dodáván s produktem IBM MQ 8.0 nebo novějším, implementuje specifikaci JMS 2.0 . Lze jej implementovat pouze na aplikační server, který je kompatibilní se systémem Java Platform, Enterprise Edition 7 (Java EE 7), a proto podporuje JMS 2.0.

Seznam certifikovaných aplikačních serverů je udržován na webových stránkách Oracle.

Implementace v produktu WebSphere Application Server Liberty

WebSphere Liberty 8.5.5 Fix Pack 6 a novější a WebSphere Application Server Liberty 9.0 a později jsou Java EE 7 certifikovanými aplikačními servery, takže je do nich lze implementovat adaptér prostředků produktu IBM MQ 8.0 nebo novější.

Produkt WebSphere Application Server Liberty má k dispozici funkci wmqJmsClient-1.1 , která umožňuje práci s adaptéry prostředků produktu JMS 1.1 a funkcí wmqJmsClient-2.0 pro povolení práce s adaptéry prostředků JMS 2.0 . Adaptér prostředku IBM MQ 8.0 nebo novější musí být implementován s funkcí wmqJmsClient-2.0 .

Informace o této konfiguraci se nacházejí ve scénáři Připojení profilu Liberty serveru WebSphere Application Server k produktu IBM MQ.

Implementace v produktu WebSphere Application Server traditional

Produkt WebSphere Application Server traditional 9.0 je dodáván s již nainstalovaným adaptérem prostředků produktu IBM MQ 9.0 . Adaptér prostředků se může připojit k libovolnému správci front spuštěnému v podporované verzi produktu IBM MQ nebo IBM WebSphere MQ. Další informace viz “Konektivita ke správcům front IBM MQ 8.0 a 9.0” na stránce 403.

Adaptér prostředků IBM MQ 9.0 nelze implementovat do dřívějších verzí produktu WebSphere Application Server, protože tyto verze nejsou Java EE 7 certifikovány.

Použití adaptéru prostředků s ostatními aplikačními servery

Pro všechny ostatní aplikační servery kompatibilní s produktem Java EE 7 mohou být problémy, ke kterým došlo po úspěšném dokončení adaptéru prostředků IBM MQ Test Verification Test (IVT), hlášeny do produktu IBM pro vyšetřování trasování produktu IBM MQ a dalších diagnostických informací o produktu IBM MQ . Pokud nelze produkt IBM MQ resource adapter IVT úspěšně spustit, jakékoli rozpoznané problémy pravděpodobně způsobí nesprávnou implementaci nebo nesprávné definice prostředků, které jsou specifické pro aplikační server, a problémy musí být zkoumány pomocí dokumentace aplikačního serveru a/nebo organizace podpory pro daný aplikační server.

Java Běhové prostředí

Běhové prostředí produktu Java (JRE), které se používá ke spuštění aplikačního serveru, musí být takové, které je podporováno klientem produktu IBM MQ 9.0 . Tato prostředí JRE jsou uvedena v seznamu Systémové požadavky pro IBM MQ. (Vyberte operační systém nebo sestavu komponenty, kterou chcete zobrazit, pak následujte odkaz **Java** , který je uveden na kartě **Podporovaný software** .)

Konektivita ke správcům front IBM MQ 8.0 a 9.0

Při připojování ke správci front IBM MQ 8.0 nebo 9.0 pomocí adaptéru prostředků produktu IBM MQ 9.0 , který byl implementován na certifikovaném aplikačním serveru Java EE 7 , je k dispozici úplný rozsah funkcí produktu JMS 2.0 . Chcete-li využívat této funkčnosti, je třeba adaptér prostředků připojit ke správci front pomocí běžného režimu poskytovatele systému zpráv produktu IBM MQ . Další informace viz **PROVIDERVERSION** neurčeno.

Konektivita k IBM WebSphere MQ 7.5 nebo starším správcům front

Je podporováno implementování adaptéru prostředků produktu IBM MQ 9.0 do certifikovaného aplikačního serveru Java EE 7 , který podporuje produkt JMS 2.0 a připojuje tento adaptér prostředků ke správci front, který spouští produkt IBM WebSphere MQ 7.5 nebo dřívější. Funkčnost, která je k dispozici, je omezena funkcemi správce front. Další informace viz ../com.ibm.mq.con.doc/q123360_.dita#q123360_.

Rozšíření MQ

Specifikace JMS 2.0 zavádí změny v tom, jak určité chování funguje. Protože produkt IBM MQ 8.0 a 9.0 implementují tuto specifikaci, došlo ke změnám v chování mezi těmito dvěma verzemi a předchozími verzemi produktu IBM WebSphere MQ. Produkt IBM MQ 8.0 a 9.0 IBM MQ classes for JMS zahrnuje podporu systémové vlastnosti Java `com.ibm.mq.jms.SupportMQExtensions` , která při nastavení na hodnotu TRUEzpůsobí, že tato dvě verze vrátí tato chování k těm, které mají IBM WebSphere MQ 7.5 nebo dřívější. Výchozí hodnota vlastnosti je FALSE.

Adaptér prostředku IBM MQ 9.0 také obsahuje vlastnost adaptéru prostředků s názvem `supportMQExtensions` , která má stejný efekt a výchozí hodnotu jako systémovou vlastnost `com.ibm.mq.jms.SupportMQExtensions` Java . This resource adapter property is set to false in the `ra.xml` by default.

Je-li nastavena vlastnost adaptéru prostředků i systémová vlastnost produktu Java , má přednost vlastnost systému.

Povšimněte si, že v rámci adaptéru prostředků, který je již implementován v produktu WebSphere Application Server traditional 9.0, je tato vlastnost automaticky nastavena na hodnotu TRUE při migraci pomoci.

Další informace viz “Vlastnost SupportMQExtensions” na stránce 294.

Obecné otázky

Prokládání relací není podporováno.
Některé aplikační servery poskytují schopnost nazývanou prokládání relací, kde lze ve více transakcích použít stejnou relaci JMS , i když je v daném okamžiku zařazena pouze do jedné relace. Adaptér prostředků produktu IBM MQ tuto schopnost nepodporuje, což může vést k následujícím problémům:

```
Pokus o vložení zprávy do fronty MQ selže s kódem příčiny 2072 (MQRC_SYNCPINT_NOT_AVAILABLE).
```

Volání metody xa_close () se nezdařila s kódem příčiny -3 (XAER_PROTO) a FDC s ID sondy AT040010 je vygenerováno na správci front produktu IBM MQ , k němuž se přistupuje z aplikačního serveru. Informace o tom, jak tuto schopnost vypnout, naleznete v dokumentaci k aplikačnímu serveru.

Specifikace Java Transaction API (JTA), jak jsou prostředky XA obnoveny pro zotavení transakce XA
Oddíl 3.4.8 specifikace JTA nedefinuje specifický mechanismus, pomocí kterého jsou prostředky XA znovu vytvořeny pro provedení zotavení transakcí XA. Jako takový je na každém správci transakcí (a tedy i na aplikačním serveru), jak se zotavují prostředky XA zapojené do transakce XA. Je možné, že pro některé aplikační servery adaptér prostředků produktu IBM MQ 9.0 neimplementuje specifické mechanizmy aplikačního serveru, které se používají k provedení zotavení transakcí XA.

Odpovídající připojení v továrně ManagedConnection
Aplikační server může vyvolat metodu matchManagedConnections na instanci továrny ManagedConnectionposkytované adaptérem prostředků produktu IBM MQ . Objekt ManagedConnection je vrácen pouze v případě, že metoda najde shodu, která odpovídá oběma argumentům **`javax.security.auth.Subject`** a **`javax.resource.spi.ConnectionRequestInfo`** , které byly předány metodě aplikačním serverem.

Omezení adaptéru prostředků produktu IBM MQ

Adaptér prostředku IBM MQ je podporován na všech platformách IBM MQ . Pokud však použijete adaptér prostředků produktu IBM MQ , některé funkce produktu IBM MQ jsou nedostupné nebo omezené.

Adaptér prostředku IBM MQ má následující omezení:

- Od IBM MQ 8.0je adaptér prostředků adaptérem prostředků Java Platform, Enterprise Edition 7 (Java EE 7), který poskytuje funkci JMS 2.0 . V důsledku toho musí být adaptér prostředků produktu IBM MQ 8.0 nebo novější instalován v certifikovaném aplikačním serveru produktu Java EE 7 nebo novější. Může se připojit v režimu transportu klienta nebo vazby k libovolnému správci front v rámci služby.
- Při spuštění uvnitř aplikačního serveru WebSphere Application Server Liberty nejsou stabilizované IBM MQ classes for Java podporovány. V jiných aplikačních serverech se IBM MQ classes for Java nedoporučuje k použití. Podrobnosti o aspektech IBM MQ classes for Java v produktu Java EEnaleznete v technické poznámce IBM Použití rozhraní Java produktu WebSphere MQ Java v prostředí J2EE/JEE Environments .
- Při spuštění v aplikačním serveru WebSphere Application Server Liberty v systému z/OSmusí být použita funkce wmqJmsClient-2.0 . Generická podpora JCA není možná pro z/OS.
- Adaptér prostředků IBM MQ nepodporuje ukončovací programy kanálu, které jsou napsány v jiných jazycích než Java.
- Když je aplikační server spuštěný, hodnota požadované vlastnosti sslFipsmusí být true pro všechny prostředky JCA nebo false pro všechny prostředky JCA . To platí i v případě, že prostředky produktu JCA nejsou souběžně používány. Pokud požadovaná vlastnost sslFipsmá odlišné hodnoty pro různé prostředky JCA , IBM MQ vydává kód příčiny MQRC_UNSUPPORTED_CIPHER_SUITE, i když není používáno připojení TLS.
- Pro aplikační server nelze určit více než jedno úložiště klíčů. Pokud jsou připojení prováděna ve více než jednom správci front, musí všechna připojení používat stejné úložiště klíčů. Toto omezení se nevztahuje na WebSphere Application Server.
- Pokud používáte tabulku CCDT (Client Channel Definition CCDT) s více než jednou vhodnou definicí kanálu pro připojení klienta, může v případě selhání adaptéru prostředku vybrat jinou definici kanálu a tudíž jiného správce front z tabulky CCDT, která by způsobila problémy při obnově transakcí. Adaptér

prostředků neprovede žádnou akci, aby zabránil použití takové konfigurace, a je vaší zodpovědností vyhnout se konfiguracím, které by mohly způsobit problémy při obnově transakcí.

- Funkčnost nového pokusu o připojení zavedená v produktu IBM WebSphere MQ 7.0.1 není podporována pro odchozí připojení při spuštění v kontejneru Java EE (EJB/Servlet). Nový pokus o připojení není podporován pro odchozí JMS , je-li adaptér použit v kontextu kontejneru JEE , bez ohledu na konfiguraci transakce nebo pro použití bez transakcí.
- Opětovné ověření, jak je definováno v sekci 9.1.9 specifikace Java EE Connector Architecture verze 1.7 , není podporováno u JMS připojení. Soubor `ra.xml` v rámci adpater prostředku IBM MQ musí mít vlastnost s názvem **`reauthentication-support`** nastavenou na hodnotu `false`. Pokus aplikačního serveru o opětovné ověření připojení JMS má za následek, že adaptér prostředků produktu IBM MQ zahodil javax.resource.spi.SecurityException s kódem zprávy MQJCA1028 .

Související informace

Určení, že pro běhové prostředí klienta MQI je použit pouze certifikovaný standard FIPS CipherSpecs

Federální standardy zpracování informací (FIPS) pro UNIX, Linux a Windows

WebSphere Application Server a adaptér prostředků IBM MQ

Adaptér prostředků produktu IBM MQ je používán aplikacemi, které provádějí systém zpráv JMS s poskytovatelem systému zpráv produktu IBM MQ v produktu WebSphere Application Server.

Důležité: Nepoužívejte adaptér prostředku IBM MQ nebo IBM WebSphere MQ s hodnotou WebSphere Application Server 6.0 nebo 6.1.

WebSphere Application Server 7.0 a WebSphere Application Server 8.0 zahrnují verzi adaptéru prostředků IBM WebSphere MQ 7.0 .

Produkt WebSphere Application Server 8.5.5 obsahuje verzi adaptéru prostředků produktu IBM WebSphere MQ 7.1 .

V 9.0.0 Produkt WebSphere Application Server traditional 9.0 obsahuje verzi adaptéru prostředků IBM MQ 9.0 . Adaptér prostředků IBM MQ 9.0 nelze implementovat do dřívějších verzí produktu WebSphere Application Server, protože tyto verze nejsou Java EE 7 certifikovány.

Chcete-li použít aplikaci JMS pro přístup k prostředkům správce front produktu IBM MQ z produktu WebSphere Application Server, použijte v produktu WebSphere Application Serverposkytovatele systému zpráv produktu IBM MQ . Poskytovatel systému zpráv produktu IBM MQ obsahuje verzi produktu IBM MQ classes for JMS. Další informace naleznete v technické poznámce Jaká verze adaptéru prostředků WebSphere MQ (RA) je dodávána se serverem WebSphere Application Server ?.

Důležité: Do své aplikace nezahrnujte žádné soubory JAR produktu IBM MQ classes for JMS nebo IBM MQ classes for Java . Výsledkem toho může být výjimka ClassCastException a může být obtížné ji udržet.

Liberty a adaptér prostředků IBM MQ

Adaptér prostředků produktu IBM MQ lze instalovat do produktu WebSphere Application Server Liberty verze WebSphere Application Server 8.5.5 Fix Pack 2 nebo novější pomocí funkce wmqJmsClient-1.1 nebo wmqJmsClient-2.0 , v závislosti na tom, kterou verzi adaptéru prostředků instalujete. Alternativně můžete, s výhradou určitých omezení, nainstalovat adaptér prostředků pomocí generické podpory produktu Java Platform, Enterprise Edition Connector Architecture (Java EE JCA).

Obecná omezení při instalaci adaptéru prostředků do produktu Liberty

Následující omezení se vztahují na adaptér prostředků při použití funkce wmqJmsClient-1.1 nebo wmqJmsClient-2.0 a také při použití generické podpory produktu JCA :

- IBM MQ classes for Java nejsou v produktu Libertypodporovány. Nesmí se používat buď s funkcí systému zpráv produktu IBM MQ Liberty , ani s obecnou podporou produktu JCA . Další informace naleznete v tématu Použití rozhraní Java produktu WebSphere MQ v prostředí J2EE/JEE Environments.

- Adaptér prostředku IBM MQ má typ transportu BINDINGS_THEN_CLIENT. Tento typ přenosu není podporován v rámci funkce systému zpráv produktu IBM MQ Liberty .
- V 9.0.0 Před IBM MQ 9.0nebyla funkce Advanced Message Security (AMS) zahrnuta do funkce systému zpráv produktu IBM MQ Liberty . Produkt AMS je však podporován s adaptérem prostředků IBM MQ 9.0 .

Omezení při použití funkcí produktu Liberty

Při použití Liberty WebSphere Application Server 8.5.5 Fix Pack 2 až WebSphere Application Server 8.5.5 Fix Pack 5 včetně, byla k dispozici pouze funkce wmqJmsClient-1.1 a lze ji použít pouze JMS 1.1 . Liberty Produkt WebSphere Application Server 8.5.5 Fix Pack 6 přidal funkci wmqJmsClient-2.0 , aby bylo možné použít příkaz JMS 2.0 .

Funkce, kterou musíte použít, závisí na tom, jakou verzi adaptéru prostředků používáte:

- Adaptér prostředku IBM WebSphere MQ 7.5.0 Fix Pack 6 a novější IBM WebSphere MQ 7.5 lze použít pouze s funkcí wmqJmsClient-1.1 .
- Adaptér prostředku IBM MQ 8.0.0 Fix Pack 3 a novější IBM MQ 8.0 lze použít pouze s funkcí wmqJmsClient-2.0 .
- Adaptér prostředků produktu IBM MQ 9.0 lze použít pouze s funkcí wmqJmsClient-2.0 .

Omezení při použití generické podpory produktu JCA

Používáte-li generickou podporu produktu JCA , platí následující omezení:

- Při použití generické podpory produktu JCA je třeba určit úroveň produktu JMS :
  - JMS 1.1 a JCA 1.6 musí být použity pouze s adaptéry prostředků IBM WebSphere MQ 7.5.0 Fix Pack 6 a novějších IBM WebSphere MQ 7.5 .
  - JMS 2.0 a JCA 1.7 musí být použity pouze s adaptéry prostředků IBM MQ 8.0.0 Fix Pack 3 a novějších IBM MQ 8.0 .
- Není možné spustit adaptér prostředků produktu IBM MQ v systému z/OS s použitím generické podpory JCA . Chcete-li spustit adaptér prostředků produktu IBM MQ v systému z/OS, musí být spuštěn s funkcí wmqJmsClient-1.1 nebo wmqJmsClient-2.0 .
- Umístění adaptéru prostředků je určeno pomocí následujícího prvku XML:

```
<resourceAdapter id="mqJms" location="${server.config.dir}/wmq.jmsra.rar">
  <classloader apiTypeVisibility="spec, ibm-api, api, third-party"/>
</resourceAdapter>
```

Důležité: Hodnota značky ID může být libovolná hodnota EXCEPT pro `wmqJms`. Pokud jako ID použijete volbu `wmqJms` , pak produkt Liberty není schopen řádně načíst adaptér prostředků. Důvodem je to, že `wmqJms` je ID, které se interně používá k odkazování na specifickou funkci pro produkt IBM MQ. Ve skutečnosti se vytváří výjimka NullPointer.

Následující příklady zobrazují některé úseky kódu ze souboru `server.xml` :

```
<!-- Enable features -->
  <featureManager>
    <feature>servlet-3.1</feature>
    <feature>jndi-1.0</feature>
    <feature>jca-1.7</feature>
    <feature>jms-2.0</feature>
  </featureManager>
```

Tip: Všimněte si použití funkcí jca-1.7 a jms-2.0 a absence funkce wmqJmsClient-2.0 .

```
<resourceAdapter id="mqJms" location="${server.config.dir}/wmq.jmsra.rar">
  <classloader apiTypeVisibility="spec, ibm-api, api, third-party"/>
</resourceAdapter>
```

Tip: Všimněte si použití `mqJms` pro ID, které je preferováno. Nepoužívejte `wmqJms`.

```
<application id="WMQHTTP" location="${server.config.dir}/apps/WMQHTTP.war"
name="WMQHTTP" type="war">
    <classloader apiTypeVisibility="spec, ibm-api, api, third-party"
classProviderRef="mqJms"/>
</application>
```

Tip: Všimněte si odkazu classloaderProviderOdkaz zpět na adaptér prostředků prostřednictvím ID `mqJms`; to znamená povolit načtení specifických tříd produktu IBM MQ .

Omezení při trasování pomocí generické podpory produktu JCA

Trasování a protokolování není integrováno do trasovacího systému Liberty . Namísto toho musí být trasování adaptéru prostředků produktu IBM MQ povoleno buď pomocí systémových vlastností produktu Java , nebo pomocí konfiguračního souboru IBM MQ classes for JMS , jak je popsáno v tématu Trasování tříd IBM MQ pro aplikace JMS. Podrobnosti o nastavení systémových vlastností produktu Java v produktu Libertynaleznete v dokumentaci produktuWebSphere Application Server Liberty.

Chcete-li například povolit trasování adaptéru prostředků IBM MQ v produktu Liberty 19.0.0.9, přidejte položku do souboru Liberty `jvm.options`:

1. Vytvořte textový soubor s názvem `jvm.options`.
2. Vložte následující volby prostředí JVM, chcete-li povolit trasování, jeden na řádek, do tohoto souboru:

   ```
   -Dcom.ibm.msg.client.commonservices.trace.status=ON
   -Dcom.ibm.msg.client.commonservices.trace.outputName=C:\Trace\MQRA-WLP_%PID%.trc
   ```

3. Chcete-li tato nastavení použít na jeden server, uložte produkt `jvm.options` na adresu:

   ```
   ${server.config.dir}/jvm.options
   ```

   Chcete-li tyto změny použít na všech Liberty, uložte `jvm.options` na:

   ```
   ${wlp.install.dir}/etc/jvm.options
   ```

   Tato akce bude platit pro všechna prostředí JVM, která nemají lokálně definovaný soubor `jvm.options` .
4. Restartujte server, abyste povolili změny.

To má za následek to, že trasování bude zapsáno do souboru trasování s názvem `MQRA-WLP_<process identifier>.trc` v adresáři `<path_to_trace_to>`.

Instalace adaptéru prostředků produktu IBM MQ

Adaptér prostředku IBM MQ je dodáván jako archivní soubor prostředku (RAR). Nainstalujte soubor RAR na aplikační server. Možná budete muset přidat adresáře do systémové cesty.

Informace o této úloze

Adaptér prostředku IBM MQ je dodáván jako archivní soubor prostředku (RAR) s názvem `wmq.jmsra.rar`. Soubor RAR obsahuje IBM MQ classes for JMS a implementaci IBM MQ rozhraní Java EE Connector Architecture (JCA).

Pokud instalujete adaptér prostředků jako součást instalace produktu IBM MQ , je produkt `wmq.jmsra.rar` nainstalován s produktem IBM MQ classes for JMS v adresáři, který je zobrazen v souboru Tabulka 61 na stránce 407.

Tabulka 61. Adresář obsahující soubor wmq.jmsra.rar pro jednotlivé platformy

Platforma	**Adresář**
UNIX and Linux	*MQ_INSTALLATION_PATH*/java/lib/jca

Tabulka 61. Adresář obsahující soubor wmq.jmsra.rar pro jednotlivé platformy (pokračování)	
Platforma	**Adresář**
IBM i	/QIBM/ProdData/mqm/java/lib/jca
Windows	*MQ_INSTALLATION_PATH*\java\lib\jca.
z/OS	*MQ_INSTALLATION_PATH*/java/lib/jca

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Chcete-li se připojit k produktu IBM MQ z aplikačního serveru, musíte použít adaptér prostředků produktu IBM MQ . V závislosti na tom, který aplikační server používáte, může být adaptér prostředků předinstalován, nebo jej možná budete muset nainstalovat sami.

Tabulka 62. Instalace adaptéru prostředků na aplikační server	
Aplikační server	**Instalován před instalací nebo je třeba jej instalovat?**
WebSphere Application Server traditional 9.0	Adaptér prostředků produktu IBM MQ 9.0 je předinstalován v rámci produktu WebSphere Application Server traditional 9.0. Proto není třeba v produktu WebSphere Application Server traditional 9.0instalovat nový adaptér prostředků.
WebSphere Application Server Liberty	WebSphere Application Server Liberty neobsahuje adaptér prostředků IBM MQ , takže jej musíte získat odděleně od Fix Central.
Jiný aplikační server Java EE	Získejte adaptér prostředků odděleně od Fix Central, jako pro WebSphere Application Server Liberty.

Procedura

- Pokud se připojujete k produktu IBM MQ z produktu WebSphere Application Server Libertynebo k jinému aplikačnímu serveru Java EE , stáhněte a nainstalujte adaptér prostředků produktu IBM MQ , jak je popsáno v tématu “Instalace adaptéru prostředků v produktu Liberty” na stránce 409.
- Linux UNIX

  Pro vázání vazeb na systémech UNIX and Linux se ujistěte, že je adresář obsahující knihovny JNI (Java Native Interface) v systémové cestě.

  Informace o umístění tohoto adresáře, který obsahuje také knihovny IBM MQ classes for JMS , viz “Konfigurace knihoven JNI (Java Native Interface)” na stránce 79.

  Windows V systému Windowsje tento adresář automaticky přidán do systémové cesty během instalace produktu IBM MQ classes for JMS.

  Tip: Jako alternativu k nastavení systémové cesty má adaptér prostředků IBM MQ vlastnost s názvem nativeLibraryPath, kterou lze použít k určení umístění knihovny JNI. Například v WebSphere Application Server Liberty by to mělo být nakonfigurováno tak, jak je zobrazeno v následujícím příkladu:

  ```
  <wmqJmsClient nativeLibraryPath="/opt/mqm/java/lib64"/>
  ```

  Transakce jsou podporovány v režimu klienta i vazby.

Instalace adaptéru prostředků v produktu Liberty

Chcete-li se připojit k produktu IBM MQ z produktu WebSphere Application Server Libertynebo do jiných aplikačních serverů Java EE , je třeba použít adaptér prostředků produktu IBM MQ . Vzhledem k tomu, že produkt Liberty neobsahuje adaptér prostředků IBM MQ , je třeba jej získat odděleně od Fix Central.

Než začnete

Poznámka: Informace v tomto tématu se nevztahují na WebSphere Application Server traditional 9.0. Adaptér prostředků produktu IBM MQ 9.0 je předinstalován v rámci produktu WebSphere Application Server traditional 9.0. Proto v tomto případě neexistuje žádný požadavek na instalaci nového adaptéru prostředků.

Před spuštěním této úlohy se ujistěte, že je na vašem počítači nainstalováno prostředí Java runtime environment (JRE) a že prostředí JRE bylo přidáno do systémové cesty.

Instalační program produktu Java , který se používá v tomto instalačním procesu, nevyžaduje spuštění jako uživatel root nebo žádný specifický uživatel. Jediným požadavkem je, aby uživatel byl spuštěn jako přístup pro zápis do adresáře, do kterého mají být soubory uloženy.

Informace o této úloze

Soubor JAR pro adaptér prostředků, který lze stáhnout z webu Fix Central , je spustitelný. Spustíte-li tento spustitelný soubor, zobrazí se licenční smlouva produktu IBM MQ , která musí být přijata. Zobrazí se dotaz na adresář, do kterého se má instalovat adaptér prostředků produktu IBM MQ . V tomto adresáři se poté nainstaluje soubor RAR adaptéru prostředků a testovací program IVT (Installation Verification Test). Můžete buď přijmout výchozí nastavení, nebo zadat jiný adresář, což může být adresář adaptérů prostředků aplikačního serveru nebo jakýkoli jiný adresář ve vašem systému. Adresář je vytvořen jako součást instalace, pokud neexistuje.

Před IBM MQ 9.0byl název souboru, který má být stažen, ve formátu *V.R.M.F*-WS-MQ-Java-InstallRA.jar, například 8.0.0.6-WS-MQ-Java-InstallRA.jar. V systému IBM MQ 9.0je formát názvu souboru *V.R.M.F*-IBM-MQ-Java-InstallRA.jar, například 9.0.0.0-IBM-MQ-Java-InstallRA.jar.

Po stažení a instalaci adaptéru prostředků jste připraveni jej nakonfigurovat v produktu WebSphere Application Server Liberty.

Postup

1. Stáhněte adaptér prostředků produktu IBM MQ z webu Fix Central:
   a) Klepněte na tlačítko **Najít produkt** a poté přidejte informace pro instalaci produktu IBM MQ do následujících polí:
      - Do pole **Selektor produktu** zadejte MQ a poté ze zobrazeného seznamu vyberte **WebSphere MQ** .
      - V poli **Instalovaná verze** klepněte na šipku a poté vyberte číslo verze ze zobrazeného seznamu, například **9.0.0.0**.
      - V poli **Platforma** klepněte na šipku a vyberte svou platformu, například **Windows 64-bitů, x86**.

      Klepněte na tlačítko **Pokračovat**.
   b) Ujistěte se, že je vybrána volba **Procházet pro opravy** a v části **Další volby dotazu**zrušte zaškrtnutí volby **Zobrazit opravy, které se týkají této verze**, a vyberte **Zobrazit opravy, které mě dostanou do této verze**, a poté klepněte na tlačítko **Pokračovat**.

      Produkt Fix Central vyhledá dostupné opravy pro vybraný produkt, verzi a platformu, například **WebSphere, WebSphere MQ (9.0.0.0, Windows 64-bitů, x86)**.
   c) Vyhledejte adaptér prostředků v zobrazeném seznamu dostupných oprav.

      Příklad:

```
release level: 9.0.0.0-IBM-MQ-Install-Java-All
9.0.0.0 MQ Resource Adapter for use with Application Servers
```

Poté klepněte na název souboru adaptéru prostředků a postupujte podle procesu stahování.

2. Spusťte instalaci zadáním následujícího příkazu z adresáře, do kterého jste soubor stáhli.

   V systému IBM MQ 9.0je formát příkazu následující:

   ```
   java -jar V.R.M.F-IBM-MQ-Java-InstallRA.jar
   ```

   kde *V.R.M.F* je číslo verze, vydání, modifikace a opravné sady a `V.R.M.F-IBM-MQ-Java-InstallRA.jar` je jméno souboru, který byl stažen z Fix Central.

   Chcete-li například instalovat adaptér prostředků produktu IBM MQ pro produkt IBM MQ 9.0.0.0, použijte následující příkaz:

   ```
   java -jar 9.0.0.0-IBM-MQ-Java-InstallRA.jar
   ```

   Poznámka: Chcete-li provést tuto instalaci, musíte mít na svém počítači nainstalováno prostředí JRE a přidat k systémové cestě.

   Když zadáte příkaz, zobrazí se následující informace:

   ```
   Než budete moci použít, extrahovat nebo instalovat produkt IBM MQ 9.0, musíte přijmout
   podmínky 1. IBM International License Agreement for Evaluation of
   Programy 2. Licenční smlouva IBM International Program License Agreement a další
   informace o licenci. Prosím, přečtěte si pozorně následující licenční smlouvy.

   Licenční smlouva je samostatně zobrazitelná pomocí
   --viewLicenseVolba dohody.
   Chcete-li nyní zobrazit licenční podmínky, stiskněte klávesu Enter. Chcete-li ji přeskočit,
   stiskněte klávesu 'x'.
   ```

3. Přezkoumejte a přijměte licenční podmínky:

   a) Chcete-li zobrazit licenci, stiskněte klávesu Enter.

      Alternativně stisknutím x přeskočíte zobrazení licence.

      Po zobrazení licence nebo bezprostředně po výběru x se zobrazí následující zpráva, která vám může sdělit, že se můžete rozhodnout pro zobrazení dalších licenčních podmínek:

      ```
      Další informace o licenci jsou samostatně viditelné pomocí
      --viewLicenseInformace o volbě.
      Stisknutím klávesy Enter zobrazíte další informace o licenci nyní, nebo 'x' pro přeskočení.
      ```

   b) Chcete-li zobrazit další licenční podmínky, stiskněte klávesu Enter.

      Alternativně stisknutím x přeskočíte zobrazení dodatečných licenčních podmínek.

      Po zobrazení dalších licenčních podmínek nebo bezprostředně po výběru x se zobrazí následující zpráva, která vás požádá o přijetí licenční smlouvy:

      ```
      Vyberete-li níže uvedenou volbu "Souhlasím", souhlasíte s podmínkami
      případně licenční smlouvy a podmínky, které nejsouIBM , pokud jsou použitelné. Pokud ne,
      Souhlasím, vyberte volbu "Nesouhlasím".

      Vyberte [ 1] Souhlasím, nebo [ 2] Nesouhlasím:
      ```

   c) Chcete-li přijmout licenční smlouvu a pokračovat výběrem instalačního adresáře, vyberte volbu 1.

      Případně, pokud vyberete volbu 2, instalace se okamžitě ukončí.

      Pokud jste vybrali volbu 1, zobrazí se následující zpráva s dotazem, zda chcete vybrat cílový instalační adresář:

      ```
      Zadejte adresář pro soubory produktu nebo ponechte pole prázdné, chcete-li přijmout výchozí
      hodnotu.
      Výchozí cílový adresář je H: \Liberty\WMQ
      Cílový adresář pro soubory produktu?
      ```

4. Určete instalační adresář pro adaptér prostředků:

   - Chcete-li instalovat adaptér prostředků do výchozího umístění, stiskněte klávesu Enter bez určení hodnoty.

- Chcete-li instalovat adaptér prostředků do jiného umístění z výchozího umístění, zadejte název adresáře, do něhož chcete adaptér prostředků instalovat, a poté stiskněte klávesu Enter.

Po instalaci souborů ve vybraném umístění se zobrazí zpráva s potvrzením, jak je zobrazeno v následujícím příkladu:

```
Extrahování souborů do H: \Liberty\WMQ \wmq
Všechny soubory produktu se úspěšně extrahovaly.
```

Během instalace se v rámci vybraného instalačního adresáře vytvoří nový adresář s názvem `wmq` a v adresáři `wmq` se pak nainstalují následující soubory:

- Testovací program ověření instalace, `wmq.jmsra.ivt`.
- Soubor RAR IBM MQ , `wmq.jmsra.rar`.

5. Konfigurujte adaptér prostředků v produktu WebSphere Application Server Liberty.

   Kroky, které je třeba provést ke konfiguraci adaptéru prostředků v produktu Liberty , jsou následující. Další informace naleznete v dokumentaci produktuWebSphere Application Server.

   a) Přidejte funkci wmqJmsClient-2.0 do souboru `server.xml` , abyste mohli povolit práci s adaptérem prostředků produktu IBM MQ 9.0 .

      Funkce, kterou přidáte (wmqJmsClient-1.1 nebo wmqJmsClient-2.0), závisí na tom, jakou verzi adaptéru prostředku jste nainstalovali. Adaptér prostředků produktu IBM MQ 9.0 musí být implementován s funkcí wmqJmsClient-2.0 . Další informace viz “Kterou verzi adaptéru prostředku použít” na stránce 401.

   b) Přidejte odkaz na soubor `wmq.jmsra.rar` , který jste nainstalovali.

   Poznámka: For Liberty versions up to WebSphere Application Server 8.5.5 Fix Pack 1, if an EJB is deployed using solely the configuration within the `ejb-jar.xml`, the version of WebSphere Application Server that the Liberty Profile is using must have APAR PM89890 applied. Tato metoda konfigurace se používá pro program pro ověření instalace adaptéru prostředků, takže tato oprava APAR je nutná k tomu, aby mohl IVT spustit.

   Příklad konfigurace pro podporu servletů a objektů MDB s operačním systémem JNDI může vypadat takto:

```
 <featureManager>
    <feature>wmqJmsClient-1.1</feature>
    <feature>servlet-3.0</feature>
    <feature>jmsMdb-3.1</feature>
    <feature>jndi-1.0</feature>
    </featureManager>

<variable name="wmqJmsClient.rar.location"
      value="H:\Liberty\WMQ\wmq\wmq.jmsra.rar"/>
```

Konfigurace adaptéru prostředků produktu IBM MQ

Chcete-li konfigurovat adaptér prostředků produktu IBM MQ , definujte různé prostředky produktu Java Platform, Enterprise Edition Connector Architecture (JCA) a volitelně i systémové vlastnosti. Chcete-li spustit program IVT (Installation Verification Test), musíte také nakonfigurovat adaptér prostředků. To je důležité, protože služba produktu IBM může tento program vyžadovat, aby označoval, že byl správně nakonfigurován aplikační server bez produktuIBM .

Než začnete

Tato úloha předpokládá, že jste již obeznámeni s produkty JMS a IBM MQ classes for JMS. Mnohé z vlastností používaných ke konfiguraci adaptéru prostředků produktu IBM MQ jsou ekvivalentní vlastnostem objektů IBM MQ classes for JMS a mají stejnou funkci.

Informace o této úloze

Každý aplikační server poskytuje svou vlastní sadu rozhraní administrace. Některé aplikační servery poskytují grafická uživatelská rozhraní pro definování prostředků produktu JCA , ale ostatní vyžadují,

aby administrátor zapsal plány implementace XML. Proto je mimo rozsah této dokumentace k dispozici informace o tom, jak nakonfigurovat adaptér prostředků IBM MQ pro každý aplikační server.

Následující kroky se proto zaměřují pouze na to, co je třeba nakonfigurovat. Informace o tom, jak nakonfigurovat adaptér prostředků produktu JCA , naleznete v dokumentaci dodávané s aplikačním serverem.

Procedura

Definujte JCA prostředky v následujících kategoriích:

- Definujte vlastnosti objektu ResourceAdapter .

  Tyto vlastnosti, které představují globální vlastnosti adaptéru prostředků, jako je například úroveň trasování diagnostiky, jsou popsány v tématu "Konfigurace pro vlastnosti objektu ResourceAdapter" na stránce 413.
- Definujte vlastnosti objektu ActivationSpec .

  Tyto vlastnosti určují, jak je objekt MDB aktivován pro příchozí komunikaci. Další informace viz "Konfigurace adaptéru prostředků pro příchozí komunikaci" na stránce 415.
- Definujte vlastnosti objektu ConnectionFactory .

  Aplikační server používá tyto vlastnosti k vytvoření objektu JMS ConnectionFactory pro odchozí komunikaci. Další informace viz "Konfigurace adaptéru prostředků pro odchozí komunikaci" na stránce 431.
- Definujte vlastnosti spravovaného cílového objektu.

  Aplikační server používá tyto vlastnosti k vytvoření objektu fronty JMS nebo objektu tématu JMS pro odchozí komunikaci. Další informace viz "Konfigurace adaptéru prostředků pro odchozí komunikaci" na stránce 431.
- Volitelné: Definujte plán implementace pro adaptér prostředků.

  Soubor RAR adaptéru prostředků produktu IBM MQ obsahuje soubor s názvem `META-INF/ra.xml`, který obsahuje deskriptor implementace pro adaptér prostředků. Tento deskriptor implementace je definován schématem XML v https://xmlns.jcp.org/xml/ns/javaee/connector_1_7.xsd a obsahuje informace o adaptéru prostředků a službách, které poskytuje. Aplikační server může také vyžadovat plán implementace pro adaptér prostředků. Tento plán implementace je specifický pro aplikační server.

Zadejte vlastnosti systému JVM podle potřeby:

- Používáte-li protokol TLS (Transport Layer Security), určete umístění souboru úložiště klíčů a souboru úložiště údajů o důvěryhodnosti jako systémové vlastnosti prostředí JVM, jako v následujícím příkladu:

  ```
  java ... -Djavax.net.ssl.keyStore=
  key_store_location
           -Djavax.net.ssl.trustStore=trust_store_location
           -Djavax.net.ssl.keyStorePassword=key_store_password
  ```

  Tyto vlastnosti nemohou být vlastnostmi objektu ActivationSpec nebo ConnectionFactory a pro aplikační server nelze určit více než jedno úložiště klíčů. Vlastnosti se vztahují na celé prostředí JVM, a mohou proto ovlivnit aplikační server, pokud jiné aplikace spuštěné na aplikačním serveru používají připojení TLS. Aplikační server může tyto vlastnosti také resetovat na různé hodnoty. Další informace o použití TLS s produktem IBM MQ classes for JMSviz "Použití TLS s IBM MQ classes for JMS" na stránce 219.
- Volitelné: Je-li to nutné, nakonfigurujte adaptér prostředků tak, aby protokolován varovné zprávy na standardní výstupní protokol aplikačního serveru.

  Protokoly adaptéru prostředků, varování a chybové zprávy používají stejný mechanismus jako produkt IBM MQ classes for JMS. Další informace naleznete v tématu Protokolování chyb pro produkt IBM MQ classes for JMS. To znamená, že při výchozím nastavení jsou zprávy odesílány do souboru s názvem `mqjms.log`. Chcete-li konfigurovat adaptér prostředků tak, aby do standardního výstupního protokolu

aplikačního serveru dodatečně protokolovací zprávy protokolovaných zpráv, nastavte následující systémovou vlastnost prostředí JVM pro aplikační server:

```
-Dcom.ibm.msg.client.commonservices.log.outputName=mqjms.log,stdout
```

Jedná se o stejnou vlastnost jako ta, která se používá k řízení trasování pro IBM MQ classes for JMS. Stejně jako u IBM MQ classes for JMSje možné použít systémovou vlastnost odkazující na soubor `jms.config` (viz “Konfigurační soubor IBM MQ classes for JMS” na stránce 81). Informace o tom, jak nastavit systémovou vlastnost prostředí JVM, najdete v dokumentaci k aplikačnímu serveru.

Konfigurujte adaptér prostředků, aby spustil test ověření instalace.

- Nakonfigurujte adaptér prostředků pro spuštění programu IVT (Installation verification test) dodaného s adaptérem prostředků produktu IBM MQ .

  Informace o tom, co je třeba nakonfigurovat, aby bylo možné spustit program IVT, najdete v tématu “Ověření instalace adaptéru prostředků” na stránce 449.

  To je důležité, protože služba IBM může vyžadovat spuštění tohoto programu, aby označoval, že byl správně nakonfigurován aplikační server jiného typu nežIBM .

  Důležité: Než budete moci spustit program, musíte nakonfigurovat adaptér prostředků.

Konfigurace pro vlastnosti objektu ResourceAdapter

Objekt ResourceAdapter zapouzdřuje globální vlastnosti adaptéru prostředků produktu IBM MQ , jako je například úroveň trasování diagnostiky. Chcete-li tyto vlastnosti definovat, použijte prostředky vašeho adaptéru prostředků, jak je popsáno v dokumentaci dodané s aplikačním serverem.

Objekt ResourceAdapter má dvě sady vlastností:

- Vlastnosti přidružené k trasování diagnostiky
- Vlastnosti přidružené k fondu připojení spravovanému adaptérem prostředků

Způsob, jakým definujete tyto vlastnosti, závisí na rozhraních administrace, které poskytuje váš aplikační server. Používáte-li WebSphere Application Server traditional, viz “Konfigurace produktu WebSphere Application Server traditional” na stránce 415 , nebo pokud používáte WebSphere Application Server Liberty, viz “Konfigurace produktu WebSphere Application Server Liberty” na stránce 415. Pro ostatní aplikační servery si prohlédněte dokumentaci k produktu pro váš aplikační server.

Další informace o definování vlastností přidružených k diagnostickému trasování najdete v tématu Trasování adaptéru prostředků IBM MQ

Adaptér prostředků spravuje interní fond připojení JMS připojení, která se používají k doručování zpráv do objektů MDB. Příkaz Tabulka 63 na stránce 413 vypíše vlastnosti objektu ResourceAdapter , který je přidružen k fondu připojení.

Tabulka 63. Vlastnosti objektu ResourceAdapter , které jsou přidruženy k fondu připojení

Název vlastnosti	**Typ**	**Výchozí hodnota**	**Popis**
maxConnections	Řetězec	50	Maximální počet připojení ke správci front IBM MQ a maximální počet implementovaných objektů MDB.
connectionConcurrency	Řetězec	1	Maximální počet objektů MDB, které mají sdílet připojení produktu JMS . Sdílení připojení není možné a tato vlastnost má vždy hodnotu 1.
Počet reconnectionRetry	Řetězec	5	Maximální počet pokusů provedených adaptérem prostředků k opětovnému připojení ke správci front produktu IBM MQ , pokud dojde k selhání připojení.

Tabulka 63. Vlastnosti objektu ResourceAdapter , které jsou přidruženy k fondu připojení (pokračování)			
Název vlastnosti	**Typ**	**Výchozí hodnota**	**Popis**
reconnectionRetryInterval	Řetězec	300 000	Doba v milisekundách, po kterou adaptér prostředků čeká před pokusem o opětovné připojení ke správci front IBM MQ .
Počet startupRetryPočet	Řetězec	0	Výchozí počet pokusů o připojení objektu MDB ke spuštění, pokud správce front není spuštěn při spuštění aplikačního serveru.
Interval startupRetryInterval	Řetězec	30 000	Výchozí doba spánku mezi pokusy o spuštění připojení (v milisekundách).
V 9.0.0 V 9.0.0 supportMQExtensions	Řetězec	ne	Přehodí chování IBM MQ JMS na chování předJMS 2.0 . Další informace viz “Vlastnost SupportMQExtensions” na stránce 294.
V 9.0.0 V 9.0.0 Cesta nativeLibrary	Řetězec	<empty>	Cesta, která má být použita k načtení knihovny JNI produktu IBM MQ pro povolení připojení režimu vazeb. Windows V systému Windows musí systémová cesta také obsahovat umístění odpovídající instalace produktu IBM MQ .

Je-li objekt MDB implementován na aplikačním serveru, vytvoří se nové připojení produktu JMS a zahájí se konverzace se správcem front za předpokladu, že maximální počet připojení určený vlastností maxConnection není překročen. Maximální počet objektů typu message-driven bean je tedy roven maximálnímu počtu připojení. Pokud počet implementovaných objektů MDB dosáhne tohoto maxima, jakýkoli pokus o implementaci jiného objektu MDB selže. Je-li objekt MDB zastaven, může být jeho připojení používáno jiným objektem MDB.

Obecně platí, že pokud se má implementovat mnoho objektů MDB, je třeba zvýšit hodnotu vlastnosti maxConnections .

Vlastnosti reconnectionRetryCount a reconnectionRetryInterval řídí chování adaptéru prostředků, když se připojení ke správci front IBM MQ nezdaří, protože došlo k selhání sítě. Dojde-li k selhání připojení, adaptér prostředků pozastaví doručování zpráv do všech objektů MDB dodaných tímto připojením po uplynutí intervalu určeného vlastností intervalu reconnectionRetry. Adaptér prostředků se pak pokusí znovu připojit ke správci front. Pokud se pokus nezdaří, adaptér prostředků provede další pokusy o opětovné připojení v intervalech určených vlastností Interval reconnectionRetry, dokud není dosažena mezní hodnota stanovená vlastností reconnectionRetryCount. Pokud se všechny pokusy nezdaří, je doručení trvale zastaveno, dokud nebudou objekty MDB restartovány ručně.

Obecně objekt ResourceAdapter nevyžaduje žádnou administraci. Chcete-li například povolit diagnostické trasování v systémech UNIX and Linux , můžete nastavit následující vlastnosti:

```
traceEnabled:    true
traceLevel:      10
```

Tyto vlastnosti nemají žádný účinek, pokud adaptér prostředků nebyl spuštěn, což je případ, například když jsou aplikace používající prostředky IBM MQ spuštěny pouze v kontejneru klienta. V této situaci můžete nastavit vlastnosti pro trasování diagnostiky jako systémové vlastnosti produktu Java Virtual Machine (JVM). Vlastnosti můžete nastavit pomocí příznaku -D u příkazu **java** , jako v následujícím příkladu:

```
java ... -DtraceEnabled=true -DtraceLevel=6
```

Není třeba definovat všechny vlastnosti objektu ResourceAdapter . Všechny vlastnosti ponechané nespecifikované vlastnosti mají své výchozí hodnoty. Ve spravovaném prostředí je lepší nesměšovat dva způsoby určení vlastností. Pokud je směšujete, mají systémové vlastnosti prostředí JVM přednost před vlastnostmi objektu ResourceAdapter .

Konfigurace produktu WebSphere Application Server traditional

Stejné vlastnosti jsou k dispozici pro adaptér prostředků v produktu WebSphere Application Server traditional, ale měly by být nastaveny na panel vlastností adaptéru prostředků (viz téma Nastavení poskytovatele JMS v dokumentaci produktu WebSphere Application Server traditional). Trasování je řízeno diagnostickým oddílem konfigurace produktu WebSphere Application Server traditional . Další informace naleznete v tématu Práce s diagnostickými poskytovateli v dokumentaci produktu WebSphere Application Server traditional .

Konfigurace produktu WebSphere Application Server Liberty

Adaptér prostředku je konfigurován pomocí prvků XML v souboru `server.xml` , jak ukazuje následující příklad:

```
<featureManager>
...
    <feature>wmqJmsClient-2.0</feature>
...
</featureManager>
    <variable name="wmqJmsClient.rar.location"
        value="F:/_rtc_wmq8005/_build/ship/lib/jca/wmq.jmsra.rar"/>
...
        <wmqJmsClient supportMQExtensions="true" logWriterEnabled="true"/>
```

Trasování je povoleno přidáním tohoto prvku XML:

```
<logging traceSpecification="JMSApi=all:WAS.j2c=all:"/>
```

Konfigurace adaptéru prostředků pro příchozí komunikaci

Chcete-li nakonfigurovat příchozí komunikaci, definujte vlastnosti jednoho nebo více objektů ActivationSpec .

Vlastnosti objektu ActivationSpec určují, jak objekt MDB (Message drive bean) přijímá zprávy produktu JMS z fronty produktu IBM MQ . Transakční chování objektu MDB je definováno ve svém deskriptoru implementace.

Objekt ActivationSpec má dvě sady vlastností:

- Vlastnosti, které slouží k vytvoření připojení produktu JMS ke správci front produktu IBM MQ
- Vlastnosti, které se používají k vytvoření spotřebitele připojení produktu JMS , který asynchronně doručuje zprávy do zadané fronty

Způsob, jakým definujete vlastnosti objektu ActivationSpec , závisí na rozhraních pro administraci poskytovaných aplikačním serverem.

Nové vlastnosti ActivationSpec v produktu JMS 2.0

Specifikace JMS 2.0 zavedla dvě nové vlastnosti ActivationSpec . Vlastnosti vyhledání connectionFactorya destinationLookup lze poskytnout s názvem JNDI spravovaného objektu, který má být použit jako předvolba pro ostatní vlastnosti ActivationSpec .

Předpokládejme například, že továrna připojení je definována v produktu JNDI a název objektu JNDI tohoto objektu je určen ve vlastnosti vyhledávání connectionFactorypro specifikaci aktivace. Všechny vlastnosti faktorie připojení definované v produktu JNDI se používají v předvolbách k vlastnostem v produktu Tabulka 64 na stránce 416.

Je-li místo určení definováno v produktu JNDI a název JNDI je nastaven ve vlastnosti destinationLookup ActivationSpec, pak se hodnoty použité v předvolbách v produktu Tabulka 65 na stránce 425použijí jako výchozí hodnoty. Další informace o způsobu použití těchto dvou vlastností naleznete v tématu "Vlastnosti ActivationSpec connectionFactorya vlastnosti destinationLookup" na stránce 428.

Vlastnosti použité k vytvoření připojení produktu JMS ke správci front produktu IBM MQ

Všechny vlastnosti v produktu Tabulka 64 na stránce 416 jsou volitelné.

Tabulka 64. Vlastnosti objektu ActivationSpec , které se používají k vytvoření připojení k produktu JMS

Název vlastnosti	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
applicationName	Řetězec	• Název třídy vyvolání, je-li k dispozici, je upraven tak, aby neměl být delší než 28 znaků. Není-li k dispozici, použije se řetězec `WebSphere MQ Client for Java` .	Název, pod kterým je aplikace registrována se správcem front. Tento název aplikace se zobrazí příkazem **`DISPLAY CONN MQSC/PCF`** (kde pole se nazývá **`APPLTAG`**). nebo v zobrazení IBM MQ Explorer **Připojení aplikace** (kde pole se nazývá **App name**).
brokerCCDurSubQueue [1]	Řetězec	• **SYSTEM.JMS.D.CC.SUBSCRIBER.QUEUE** • Název fronty.	Název fronty, z níž spotřebitel připojení přijímá zprávy trvalého odběru
brokerCCSubFronta [1]	Řetězec	• **SYSTEM.JMS.ND.CC.SUBSCRIBER.QUEUE** • Název fronty.	Název fronty, z níž spotřebitel připojení přijímá zprávy netrvalého odběru
brokerControlFronta [1]	Řetězec	• **SYSTEM.BROKER.CONTROL.QUEUE** • Název fronty.	Název řídicí fronty zprostředkovatele
brokerQueueManager [1]	Řetězec	• **"" (prázdný řetězec)** • Název správce front	Název správce front, v němž je zprostředkovatel spuštěn.
brokerSubFronta [1]	Řetězec	• **SYSTEM.JMS.ND.SUBSCRIBER.QUEUE** • Název fronty.	Název fronty, z níž spotřebitel zpráv netrvalých zpráv přijímá zprávy
brokerVersion [1]	Řetězec	• **neuvedený** -Po migraci zprostředkovatele z V6 do V7nastavte tuto vlastnost tak, aby záhlaví RFH2 již nebyla použita. Po migraci tato vlastnost již není relevantní. • **V1** -Chcete-li použít broker.This IBM MQ publish/subscribe, hodnota je výchozí hodnotou, pokud je funkce TRANSPORT nastavena na BIND nebo CLIENT. • **V2** -Chcete-li použít zprostředkovatele IBM Integration Bus v nativním režimu. Tato hodnota je výchozí hodnota, je-li hodnota TRANSPORT nastavena na DIRECT nebo DIRECTHTTP.	Verze používaného zprostředkovatele

Tabulka 64. Vlastnosti objektu ActivationSpec , které se používají k vytvoření připojení k produktu JMS (pokračování)

Název vlastnosti	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
ccdtURL	Řetězec	• **null** • Uniform resource locator (adresa URL)	Adresa URL, která identifikuje název a umístění souboru obsahujícího tabulku CCDT (Client Channel Definition CCDT) a určuje, jak lze k souboru přistupovat
CCSID	Řetězec	• **819** • Identifikátor kódové sady znaků podporovaný virtuálním počítačem Java (JVM)	Identifikátor kódové sady znaků pro připojení
kanál	Řetězec	• **SYSTEM.DEF.SVRCONN** • Název kanálu MQI	Název kanálu MQI, který má být použit
cleanupInterval [1]	celé číslo	• **3 600 000** • Kladné celé číslo	Interval v milisekundách mezi spuštěními obslužného programu čištění publikování/odběru na pozadí
cleanupLevel [1]	Řetězec	• **Bezpečný** • ŽÁDNÉ • velká • Vynutit • NONDUR	Úroveň vyčištění pro úložiště odběrů založené na zprostředkovateli
clientID	Řetězec	• **null** • Identifikátor klienta	Identifikátor klienta pro připojení
cloneSupport	Řetězec	• **VYPNUTÉ** -V daném okamžiku může být spuštěna pouze jedna instance trvalého odběratele tématu. • ENABLED-Dvě nebo více instancí stejného odběratele trvalého tématu mohou být spuštěny souběžně, ale každá instance musí být spuštěna v samostatném virtuálním počítači Java (JVM).	Určuje, zda mohou být dvě nebo více instancí stejného odběratele trvalého tématu spuštěny souběžně.

Tabulka 64. Vlastnosti objektu ActivationSpec , které se používají k vytvoření připojení k produktu JMS (pokračování)

Název vlastnosti	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
Vyhledání connectionFactory	Řetězec	• **null** • Název rozhraní JNDI pro objekt ConnectionFactory	Je-li tato vlastnost nastavena, vyhledá objekt ActivationSpec objekt JMS ConnectionFactory s uvedeným názvem rozhraní JNDI v oboru názvů rozhraní JNDI aplikačního serveru a poté použije vlastnosti tohoto objektu k vytvoření připojení produktu JMS ke správci front produktu IBM MQ , s jednou výjimkou. Jedinou vlastností objektu ActivationSpec , která bude použita při vytváření připojení JMS , je clientID. Další informace viz “Vlastnosti ActivationSpec connectionFactorya vlastnosti destinationLookup” na stránce 428.
ConnectionNameList	Řetězec	• **localhost (1414)** • Řetězec složený z položek oddělených čárkami, kde každá položka má formát: `HOSTNAME(PORT)` kde *HOSTNAME* je buď název DNS, nebo adresa IP.	Seznam názvů připojení TCP/IP použitých pro příchozí komunikace. Je-li tato volba zadána, **connectionNameList** nahrazuje vlastnosti **hostname** a **port** . Tato vlastnost se používá k opětovnému připojení ke správcům front s více instancemi. **connectionNameList** je podobný ve tvaru **localAddress**, ale nesmí být zaměňován s ním. Parametr **localAddress** určuje charakteristiky lokální komunikace, zatímco **connectionNameList** určuje, jak se má dostat ke vzdálenému správci front.
FAILIFQUIESCE	Logická hodnota	• **ano** • ne	Bez ohledu na to, zda volání určitých metod selže, pokud je správce front ve stavu uvedení do klidového stavu

Tabulka 64. Vlastnosti objektu ActivationSpec , které se používají k vytvoření připojení k produktu JMS (pokračování)

Název vlastnosti	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
headerCompression	Řetězec	• **Není** • SYSTEM-Komprese hlavičky zprávy RLE se provádí	Seznam technik, které lze použít pro kompresi dat záhlaví na připojení
hostName	Řetězec	• **lokální hostitel** • Název hostitele • Adresa IP	Název hostitele nebo adresa IP systému, v němž je správce front umístěn. Vlastnosti **hostname** a **port** jsou nahrazovány vlastností **connectionNameList** , je-li zadána.
localAddress	Řetězec	• **null** • Řetězec ve formátu: `[ host_name ][(low_port [, high_port ])]` kde *název_hostitele* je název hostitele nebo adresa IP, *low_port* a *high_port* jsou čísla portů TCP a hranaté závorky označují volitelnou komponentu.	Pro připojení ke správci front tato vlastnost určuje jednu z následujících možností nebo obě tyto akce: • Lokální síťové rozhraní, které má být použito • Lokální port, nebo rozsah lokálních portů, které se mají použít **localAddress** je podobný ve tvaru **connectionNameList**, ale nesmí být zaměňován s ním. Parametr **localAddress** určuje charakteristiky lokální komunikace, zatímco **connectionNameList** určuje, jak se má dostat ke vzdálenému správci front.
messageCompression	Řetězec	• **Není** • Seznam jedné nebo více následujících hodnot oddělených prázdnými znaky: RLE ZLIBFAST ZLIBHIGH	Seznam technik, které lze použít pro kompresi dat zpráv na připojení
messageRetention [1]	Logická hodnota	• **true** -Nepožadované zprávy zůstanou ve vstupní frontě. • false-Nepožadované zprávy jsou řešeny podle jejich dispozičních voleb	Zda má spotřebitel připojení uchovávat nežádoucí zprávy ve vstupní frontě

Tabulka 64. Vlastnosti objektu ActivationSpec , které se používají k vytvoření připojení k produktu JMS (pokračování)

Název vlastnosti	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
messageSelection [1]	Řetězec	• **client** • BROKER	Určuje, zda je výběr zpráv prováděn produktem IBM MQ classes for JMS nebo zprostředkovatelem. Výběr zpráv zprostředkovatelem není podporován, je-li hodnota brokerVersion nastavena na hodnotu 1.
heslo	Řetězec	• **null** • Heslo	Výchozí heslo, které má být použito při vytváření připojení ke správci front
pollingInterval [1]	celé číslo	• **5000** • Libovolné kladné celé číslo	Pokud každý modul listener pro zprávy v rámci relace nemá ve frontě žádnou odpovídající zprávu, jedná se o maximální interval v milisekundách, který uplyne předtím, než se každý modul listener pro zprávy znovu pokusí získat zprávu z příslušné fronty. Pokud se často stává, že pro žádný z listenerů zpráv v rámci relace není k dispozici žádná vhodná zpráva, zvažte zvýšení hodnoty této vlastnosti. Tato vlastnost je relevantní pouze v případě, že hodnota TRANSPORT má hodnotu BIND nebo CLIENT.
Port	celé číslo	• **1414** • Číslo portu TCP	Port, na kterém správce front naslouchá. Vlastnosti **hostname** a **port** jsou nahrazovány vlastností **connectionNameList** , je-li zadána.
providerVersion	řetězec	• **nespecifikováno** • Řetězec v jednom z následujících formátů – V.R.M.F – V.R.M – V.R – V kde V, R, M a F jsou celočíselné hodnoty větší než nebo rovné nule.	Verze, vydání, úroveň modifikace a opravná sada správce front, ke kterému má objekt MDB v úmyslu se připojit.

Tabulka 64. Vlastnosti objektu ActivationSpec , které se používají k vytvoření připojení k produktu JMS (pokračování)

Název vlastnosti	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
queueManager	Řetězec	• **"" (prázdný řetězec)** • Název správce front	Název správce front, ke kterému se má připojit
receiveExit [3]	Řetězec	• **null** • Řetězec obsahující jednu nebo více položek oddělených čárkami, kde každá položka představuje úplný název třídy, která implementuje rozhraní produktu IBM MQ classes for Java , MQReceiveExit .	Identifikuje uživatelský program pro příjem kanálu nebo posloupnost ukončovacích ukončovacích programů, které mají být spouštěny za sebou.
receiveExitInit	Řetězec	• **null** • Řetězec obsahující jednu nebo více položek dat uživatele oddělených čárkami	Uživatelská data, která jsou předána uživatelským programům přijetí kanálu při jejich volání
rescanInterval [1]	celé číslo	• **5000** • Libovolné kladné celé číslo	Pokud spotřebitel zpráv v doméně typu point-to-point používá selektor zpráv k výběru zpráv, které chce přijímat, produkt IBM MQ classes for JMS prohledá frontu IBM MQ pro vhodné zprávy v posloupnosti určené atributem **MsgDeliverySequence** fronty. Pokud produkt IBM MQ classes for JMS najde vhodnou zprávu a doručí ji spotřebiteli, produkt IBM MQ classes for JMS obnoví hledání další vhodné zprávy z aktuální pozice ve frontě. Produkt IBM MQ classes for JMS pokračuje v hledání fronty tímto způsobem, dokud nedosáhne konce fronty, nebo dokud nevyprší časový interval v milisekundách určený hodnotou této vlastnosti. V každém případě se IBM MQ classes for JMS vrátí na začátek fronty, aby pokračoval v hledání, a začne nový časový interval.
securityExit [3]	Řetězec	• **null** • Úplný název třídy, která implementuje rozhraní IBM MQ classes for Java , MQSecurityExit .	Identifikuje uživatelský program zabezpečení kanálu

Tabulka 64. Vlastnosti objektu ActivationSpec , které se používají k vytvoření připojení k produktu JMS (pokračování)

Název vlastnosti	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
Init securityExit	Řetězec	• **null** • Řetězec uživatelských dat	Uživatelská data, která jsou předána do uživatelského programu zabezpečení kanálu, je-li volán
sendExit [3]	Řetězec	• **null** • Řetězec obsahující jednu nebo více položek oddělených čárkami, kde každá položka představuje úplný název třídy, která implementuje rozhraní IBM MQ classes for Java , MQSendExit .	Identifikuje uživatelský program odeslání kanálu nebo posloupnost uživatelských ukončovacích programů, které mají být spuštěny po sobě.
SENDEXITINIT	Řetězec	• **null** • Řetězec obsahující jednu nebo více položek dat uživatele oddělených čárkami	Uživatelská data, která jsou předána uživatelským programům odeslání kanálu, když jsou volány
SHARECONVALLOWED	Logická hodnota	• **NO** -Připojení klienta nemůže sdílet svůj soket. • YES-Připojení klienta může sdílet svůj soket.	Zda může připojení klienta sdílet svůj soket s dalšími připojeními JMS nejvyšší úrovně ze stejného procesu ke stejnému správci front, pokud se definice kanálu shodují
sparseSubscriptions [1]	Logická hodnota	• **false** -Odběry přijímají časté odpovídající zprávy. • true-Odběry přijímají odpovídající zprávy zřídka. Tato hodnota vyžaduje, aby byla fronta odběru otevřena pro procházení.	Ovládá zásadu načítání zpráv objektu TopicSubscriber
Úložiště sslCert	Řetězec	• **null** • Řetězec jedné nebo více adres URL LDAP oddělených mezerami. Každá adresa URL protokolu LDAP má tento formát: `ldap://host_name [: port ]` kde *název_hostitele* je název hostitele nebo adresa IP, *port* je číslo portu TCP a hranaté závorky označují volitelnou komponentu.	Servery LDAP (Lightweight Directory Access Protocol), které uchovávají seznamy zrušených certifikátů (CRL) pro použití v připojení TLS
SSLCIPHERSUITE	Řetězec	• **null** • Název sady CipherSuite	Sada CipherSuite , která má být použita pro připojení TLS
sslFipsPovinné [2]	Logická hodnota	• **ne** • ano	Zda musí připojení TLS používat sadu CipherSuite , kterou podporuje poskytovatel prostředí JSSE FIPS produktu IBM Java (IBMJSSEFIPS)

Tabulka 64. Vlastnosti objektu ActivationSpec , které se používají k vytvoření připojení k produktu JMS (pokračování)

Název vlastnosti	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
SSLPEERNAME	Řetězec	• **null** • Šablona pro rozlišující názvy	Pro připojení TLS se jedná o šablonu, která se používá ke kontrole rozlišujícího názvu v digitálním certifikátu poskytovaného správcem front
SSLRESETCOUNT	celé číslo	• **0** • Celé číslo v rozsahu 0 až 999 999 999.	Celkový počet bajtů odeslaných a přijatých pomocí připojení TLS, než budou znovu vyjednány tajné klíče použité TLS
Továrna sslSocket	Řetězec	Řetězec představující plně kvalifikovaný název třídy třídy poskytující implementaci rozhraní javax.net.ssl.SSLSocketFactory . Volitelně včetně argumentu, který má být předán do metody konstruktoru, uzavřený v závorkách.	Veškerá spojení vytvořená v oboru administrovaného objektu používají sokety získané z této implementace rozhraní SSLSocketFactory .
statusRefreshInterval [1]	celé číslo	• **60000** • Libovolné kladné celé číslo	Interval (v milisekundách) mezi aktualizacemi transakce s dlouhou dobou zpracování, která zjišťuje, zda odběratel ztratil připojení ke správci front. Tato vlastnost je relevantní pouze v případě, že má **subscriptionStore** hodnotu QUEUE.
subscriptionStore [1]	Řetězec	• **BROKER** • MIGRATE • QUEUE	Určuje, kam IBM MQ classes for JMS ukládá trvalá data o aktivních odběrech

Tabulka 64. Vlastnosti objektu ActivationSpec , které se používají k vytvoření připojení k produktu JMS (pokračování)

Název vlastnosti	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
transportType	Řetězec	• **client** • Vazby • BINDINGS_THEN_CLIENT	Určuje, zda připojení ke správci front používá režim klienta nebo režim vazeb. Je-li zadána hodnota `BINDINGS_THEN_CLIENT` , adaptér prostředků se nejprve pokusí vytvořit připojení v režimu vazeb. Pokud se pokus o připojení nezdaří, adaptér prostředků se pokusí vytvořit připojení v režimu klienta. z/OS Pokud byla specifikace aktivace spuštěná na systému WebSphere Application Server for z/OS konfigurována tak, aby používala režim přenosu `BINDINGS_THEN_CLIENT` a dříve zavedené připojení je přerušeno, provede se pokus o opětovné připojení ze specifikace aktivace při prvním pokusu o použití režimu přenosu `BINDINGS` . Pokud se pokus o připojení režimu přenosu `BINDINGS` nezdaří, specifikace aktivace se následně pokusí o připojení v režimu transportu `CLIENT` .
jméno uživatele	Řetězec	• **null** • Jméno uživatele	Výchozí jméno uživatele, které má být použito při vytváření připojení ke správci front
wildcardFormat	Řetězec	• CHAR-Rozlišuje se pouze znakové zástupné znaky, jak je použito ve zprostředkovateli verze 1 • **TOPIC** -Rozpoznává pouze zástupné znaky na úrovni tématu, jak je použito ve verzi 2 zprostředkovatele.	Která verze syntaxe zástupných znaků se má použít

Notes:

1. Tuto vlastnost lze použít s produktem IBM MQ classes for JMS v produktu IBM WebSphere MQ 7.0. Neovlivní aplikaci připojenou ke správci front IBM WebSphere MQ 7.0 , pokud není vlastnost **`providerVersion`** nastavena na číslo verze menší než 7.

2. Důležité informace o použití vlastnosti sslFipsRequired naleznete v příručce “Omezení adaptéru prostředků produktu IBM MQ” na stránce 404.
3. Informace o tom, jak nakonfigurovat adaptér prostředků tak, aby mohl najít uživatelskou proceduru, najdete v tématu “Konfigurace produktu IBM MQ classes for JMS pro použití uživatelských procedur kanálu” na stránce 250.

Vlastnosti použité k vytvoření spotřebitele připojení JMS

Poznámka: Hodnoty **destination** a **destinationType** musí být explicitně definovány. Všechny ostatní vlastnosti v produktu Tabulka 65 na stránce 425 jsou volitelné.

Tabulka 65. Vlastnosti objektu ActivationSpec , které se používají k vytvoření spotřebitele připojení JMS

Název vlastnosti	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
cíl	Řetězec	Název místa určení	Cíl, ze kterého mají být přijímány zprávy. Vlastnost **useJNDI** určuje, jak je hodnota této vlastnosti interpretována.
destinationLookup	Řetězec	• **null** • Název rozhraní JNDI pro cílový objekt	Je-li tato vlastnost nastavena, vyhledá objekt ActivationSpec objekt cíle JMS s uvedeným názvem rozhraní JNDI v oboru názvů rozhraní JNDI aplikačního serveru a poté použije vlastnosti tohoto objektu k vytvoření spotřebitele připojení produktu JMS , a to podle dalších vlastností uvedených v objektu ActivationSpec. Další informace viz “Vlastnosti ActivationSpec connectionFactorya vlastnosti destinationLookup” na stránce 428.
destinationType	Řetězec	• javax.jms.Queue • javax.jms.Topic	Typ místa určení, fronty nebo téma
maxMessages	celé číslo	• **1** • Kladné celé číslo	Maximální počet zpráv, které mohou být přiřazeny k relaci serveru najednou. Pokud specifikace aktivace doručuje zprávy do objektu MDB v rámci transakce XA, použije se hodnota 1 bez ohledu na nastavení této vlastnosti.
Hloubka maxPool	celé číslo	• **10** • Kladné celé číslo	Maximální počet relací serveru ve fondu relací serveru používaných odběratelem připojení
messageSelector	Řetězec	• **null** • Výraz selektoru zpráv SQL92 .	Výraz selektoru zpráv určující, které zprávy mají být doručeny

Tabulka 65. Vlastnosti objektu ActivationSpec , které se používají k vytvoření spotřebitele připojení JMS (pokračování)

Název vlastnosti	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
nonASFTimeout	celé číslo	• **0** • Kladné celé číslo	Kladná hodnota označuje, že se používá doručení non-ASF. Hodnota je čas (v milisekundách), po který požadavek na získání čeká na zprávy, které možná ještě nedorazily (get with wait call). Předvolená hodnota 0 označuje, že se použije funkce ASF doručení. Tento parametr je platný, pokud: • Aplikace běží na systému WebSphere Application Server 7.0 nebo pozdější. • Aplikace je spuštěna v produktu WebSphere Application Server Liberty s použitím příslušné úrovně funkce Klient wmqJms. Další informace viz “Liberty a adaptér prostředků IBM MQ” na stránce 405.
nonASFRollbackPovoleno	Logická hodnota	• **false** -Zpráva se spotřebuje i v případě, že objekt MDB selže. • true-Selhání v rámci objektu MDB způsobí vrácení zprávy zpět do fronty.	Určuje, zda má být doručování zpráv v rámci synchronizačního bodu IBM MQ , je-li objekt MDB mimo transakci. Ignoruje se, pokud je objekt MDB zpracovávána, nebo je-li **nonASFTimeout** nastaveno na 0.
poolTimeout	celé číslo	• **300000** • Kladné celé číslo	Doba (v milisekundách), po kterou je nepoužívaná relace serveru zadržena ve fondu relací serveru před zavřením z důvodu nečinnosti
READAHEADALLOWED	celé číslo	• **DESTINATION** -Určení toho, zda je dopředné čtení povoleno odkazem na definici fronty nebo tématu. • DISABLED-Čtení napřed není povoleno. • ENABLED-Čtení napřed je povoleno. • QUEUE-Určení, zda je dopředné čtení povoleno s odkazem na definici fronty. • TOPIC-Určení, zda je dopředné čtení povoleno s odkazem na definici tématu.	Zda má objekt MDB povoleno používat dopředné čtení k získání přechodných zpráv z místa určení do interní vyrovnávací paměti, než je přijme.

Tabulka 65. Vlastnosti objektu ActivationSpec , které se používají k vytvoření spotřebitele připojení JMS (pokračování)

Název vlastnosti	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
readAheadClosePolicy	celé číslo	• **ALL** -Všechny zprávy v interní vyrovnávací paměti dopředného čtení jsou doručovány do objektu MDB před zastavením. • CURRENT-Pouze aktuální vyvolání objektu typu message-driven bean může potenciálně zanechat zprávy v interní vyrovnávací paměti dopředného čtení, které se pak vyřadí.	Co se stane se zprávami v interní vyrovnávací paměti pro čtení napřed, je-li objekt MDB zastaven administrátorem.
receiveCCSID	celé číslo	• **0** -použít prostředí JVM `Charset.defaultCharset` • 1208- UTF-8 • Podporovaný identifikátor kódované znakové sady	Vlastnost místa určení, která nastavuje cílové CCSID pro převod zpráv správce front. Hodnota je ignorována, pokud parametr **`receiveConversion`** není nastaven na hodnotu `QMGR`.
receiveConversion	Řetězec	• **CLIENT_MSG** • QMGR	Cílová vlastnost, která určuje, zda bude převod dat prováděn správcem front.
sharedSubscription	Logická hodnota	• **False** -Objekt MDB by neměl otevřít odběr jako sdílený odběr. • True-MDB by mělo otevřít odběr jako sdílený odběr (s pravidly, která JMS 2.0 implikuje, viz specifikace JMS 2.0 na adrese Java.net).	Řídí, jak je objekt MDB řízen ze sdíleného odběru. Další informace o tom, jak používat tuto vlastnost, viz "Příklady toho, jak definovat vlastnost sharedSubscription" na stránce 430.
startTimeout	celé číslo	• **10 000** • Kladné celé číslo	Doba v milisekundách, po jejímž uplynutí musí být doručení zprávy do objektu MDB spuštěno po naplánování práce na doručení zprávy. Pokud tento čas uplyne, zpráva se vrátí zpět do fronty.
subscriptionDurability	Řetězec	• **NonDurable** -Netrvalý odběr se používá k doručování zpráv do objektu MDB, který je odběratelem tématu. • Trvalý-odběr je používán k doručování zpráv do objektu MDB, který je odběratelem tématu.	Určuje, zda je k doručování zpráv do objektu MDB, který je odběratelem tématu, použit trvalý nebo trvalý odběr.
subscriptionName	Řetězec	• **"" (prázdný řetězec)** • Název odběru	Název trvalého odběru

Tabulka 65. Vlastnosti objektu ActivationSpec , které se používají k vytvoření spotřebitele připojení JMS (pokračování)			
Název vlastnosti	**Typ**	**Platné hodnoty (výchozí hodnota tučně)**	**Popis**
useJNDI	Logická hodnota	• **false** -Vlastnost s názvem destination je interpretována jako název fronty IBM MQ nebo tématu. • true-Vlastnost s názvem destination je interpretována jako název objektu javax.jms.Queue nebo objektu javax.jms.Topic v oboru názvů JNDI aplikačního serveru.	Určuje, jak je interpretována hodnota vlastnosti s názvem destination **Poznámka:** Tato vlastnost je v produktu IBM MQ 9.0zamítnuta. Místo toho by se měla použít vlastnost destinationLookup .

Konflikty vlastností a závislosti

Objekt ActivationSpec může mít konfliktní vlastnosti. Můžete například zadat vlastnosti TLS pro připojení v režimu vazeb. V tomto případě je chování určeno typem transportu a doménou systému zpráv, což je buď dvoubodové spojení, nebo publikování/odběr určené vlastností **destinationType** . Všechny vlastnosti, které nejsou použitelné pro zadaný typ transportu nebo doménu systému zpráv, se budou ignorovat.

Pokud definujete vlastnost, která vyžaduje, aby byly definovány jiné vlastnosti, ale tyto další vlastnosti nedefinujete, objekt ActivationSpec vyvolá výjimku InvalidPropertyException při volání metody validate () během implementace objektu MDB. Výjimka se nahlašte administrátorovi aplikačního serveru způsobem, který závisí na aplikačním serveru. Pokud například nastavíte vlastnost subscriptionDurability na Trvalý, což značí, že chcete používat trvalé odběry, musíte také definovat vlastnost **subscriptionName** .

Pokud jsou definovány vlastnosti nazývané **ccdtURL** a **channel** , dojde k výjimce výjimky InvalidProperty. Definujete-li však pouze vlastnost **ccdtURL** , ponechají vlastnost s názvem **channel** s její výchozí hodnotou SYSTEM.DEF.SVRCONNnení vygenerována žádná výjimka a tabulka definic kanálů klienta identifikovaná pomocí vlastnosti **ccdtURL** se používá ke spuštění připojení JMS .

Vlastnosti ActivationSpec connectionFactorya vlastnosti destinationLookup

Tyto dvě vlastnosti lze použít k určení názvů JNDI objektů ConnectionFactory a cílových objektů, které se použijí přednostně k vlastnostem objektu ActivationSpec , jak je definováno v Tabulka 64 na stránce 416 a Tabulka 65 na stránce 425.

Je důležité si povšimnout následujících bodů, které popisují, jak tyto vlastnosti fungují v detailech.

Vyhledání connectionFactory
ConnectionFactory , která je vyhledána z produktu JNDI , se používá jako zdroj vlastností vypsaných v Tabulka 64 na stránce 416. Objekt ConnectionFactory není použit ke skutečnému vytvoření žádných připojení JMS , jsou dotazovány pouze vlastnosti objektu. Tyto vlastnosti z objektu ConnectionFactory potlačí všechny vlastnosti, které jsou definovány na ActivationSpec. Existuje jedna výjimka. Má-li vlastnost ActivationSpec nastavenou vlastnost **ClientID** , hodnota této vlastnosti potlačí hodnotu zadanou v objektu ConnectionFactory. Důvodem je skutečnost, že běžný scénář používá jedinou ConnectionFactory s více ActivationSpecs. To zjednodušuje administraci. Specifikace JMS 2.0 však uvádí, že každé JMS připojení vytvořené z ConnectionFactory by mělo mít jedinečnou **ClientID**. Kvůli tomu je třeba, aby funkce ActivationSpecs měla schopnost potlačit jakoukoli hodnotu nastavenou v objektu ConnectionFactory. Není-li v parametru ActivationSpecnastavena žádná hodnota **ClientID** , použije se libovolná hodnota z továrny připojení.

destinationLookup
Vlastnosti **Destination** a **UseJndi** jsou definovány na ActivationSpec. Je-li příznak **UseJndi** nastaven na hodnotu `true`, bude text zadaný ve vlastnosti místa určení považován za název JNDI a cílový objekt s názvem JNDI se bude vyhledávat z produktu JNDI.

Vlastnost destinationLookup se chová přesně stejným způsobem. Pokud byla nastavena, bude objekt místa určení s názvem JNDI zadaným vlastností vyhledán v produktu JNDI. Tato vlastnost má přednost před vlastností **useJNDI** .

Vlastnost useJNDI je zamítnuta na IBM MQ 9.0 , protože vlastnost **destinationLookup** je specifikace JMS 2.0 ekvivalentní k provedení stejné funkce.

Vlastnosti ActivationSpec bez ekvivalentů v produktu IBM MQ classes for JMS

Většina vlastností objektu ActivationSpec je ekvivalentní vlastnostem objektů IBM MQ classes for JMS nebo parametrů metod IBM MQ classes for JMS . Nicméně tři vlastnosti ladění a jedna vlastnost použitelnosti nemají v produktu IBM MQ classes for JMSžádné ekvivalenty:

startTimeout
Doba (v milisekundách), po kterou správce Work Manager aplikačního serveru čeká na zpřístupnění prostředků poté, co adaptér prostředků naplánuje doručení zprávy do objektu typu message-driven bean. Pokud tento čas uplyne před doručením zprávy, dojde k vypršení časového limitu pracovního objektu, zpráva je vrácena do fronty a adaptér prostředků se pak může pokusit o doručení zprávy znovu. Varování se zapíše do diagnostického trasování, pokud je povoleno, ale jinak neovlivní proces doručování zpráv. Tato podmínka se může vyskytnout pouze v době, kdy aplikační server vykazuje velmi vysoké zatížení. Pokud se stav vyskytuje pravidelně, zvažte zvýšení hodnoty této vlastnosti tak, aby správci Work Manager mohl naplánovat doručení zprávy déle.

Hloubka maxPool
Maximální počet relací serveru ve fondu relací serveru používaných spotřebitelem připojení. Je-li vytvořena relace serveru, spustí konverzaci se správcem front. Spotřebitel připojení používá relaci serveru k doručení zprávy do objektu MDB. Větší hloubka fondu umožňuje souběžné doručení více zpráv v situacích s vysokým objemem dat, ale používá více prostředků aplikačního serveru. Pokud má být implementováno mnoho objektů MDB, zvažte zvýšení hloubky fondu, aby se zachovalo zatížení na aplikačním serveru na spravovatelné úrovni. Každý spotřebitel připojení používá svůj vlastní fond relací serveru, takže tato vlastnost nedefinuje celkový počet relací serveru dostupných pro všechny spotřebitele připojení.

poolTimeout
Doba (v milisekundách), po kterou je nepoužívaná relace serveru zadržena ve fondu relací serveru před zavřením z důvodu nečinnosti. Přechodný nárůst pracovní zátěže zpráv způsobí vytvoření dalších relací serveru za účelem rozdělení zátěže, ale po návratu pracovní zátěže zpráv do normálu zůstanou další serverové relace ve fondu zachovány a nebudou použity.

Pokaždé, když se použije relace serveru, je označena časovým razítkem. Periodicky průnikový podproces kontroluje, zda byla každá relace serveru použita ve lhůtě určené touto vlastností. Pokud nebyla relace serveru použita, je zavřena a odebrána z fondu relací serveru. Je možné, že relace serveru není zavřena bezprostředně po uplynutí zadané doby. Tato vlastnost představuje minimální období nečinnosti před odebráním.

useJNDI
Popis této vlastnosti viz Tabulka 65 na stránce 425.

Implementace objektu MDB

Chcete-li implementovat objekt MDB, nejprve definujte vlastnosti objektu ActivationSpec a určete vlastnosti, které objekt MDB vyžaduje. Následující příklad je typickou sadou vlastností, které můžete explicitně definovat:

```
channel:           SYSTEM.DEF.SVRCONN
destination:       SYSTEM.DEFAULT.LOCAL.QUEUE
destinationType:   javax.jms.Queue
hostName:          192.168.0.42
messageSelector:   color='red'
port:              1414
queueManager:      ExampleQM
transportType:     CLIENT
```

Aplikační server používá tyto vlastnosti k vytvoření objektu ActivationSpec , který je poté přidružen k objektu MDB. Vlastnosti objektu ActivationSpec určují, jak jsou zprávy doručovány do objektu MDB. Implementace objektu MDB se nezdaří, pokud objekt MDB vyžaduje distribuované transakce, ale adaptér prostředků nepodporuje distribuované transakce. Informace o tom, jak instalovat adaptér prostředků tak, aby byly podporovány distribuované transakce, najdete v tématu “Instalace adaptéru prostředků produktu IBM MQ” na stránce 407.

Pokud více než jeden objekt MDB přijímá zprávy ze stejného cíle, odešle se zpráva odeslaná v doméně typu point-to-point pouze jedním objektem typu message-driven bean, a to i v případě, že jsou jiné objekty MDB vhodné k přijetí zprávy. Konkrétně, pokud dva objekty MDB používají různé selektory zpráv a příchozí zpráva odpovídá oběma selektorům zpráv, obdrží zprávu pouze jeden z objektů MDB. Objekt MDB vybraný pro příjem zprávy není definován a nelze se spolehnout na konkrétní objekt MDB, který je příjemcem zprávy. Zprávy odeslané v doméně publikování/odběru jsou přijímány všemi vhodnými objekty typu message-driven bean.

Za určitých okolností může být zpráva doručená do objektu MDB vrácena do fronty produktu IBM MQ . Tento návrat se může stát například tehdy, je-li zpráva doručena v rámci pracovní jednotky, která je poté odvolána. Zpráva, která je odvolána, je doručena znovu, ale špatně formátovaná zpráva může opakovaně způsobit selhání objektu typu message-driven bean, a proto ji nelze doručit. Takováto zpráva se nazývá nezpracovatelná zpráva. Produkt IBM MQ můžete nakonfigurovat tak, aby produkt IBM MQ classes for JMS automaticky přesunoval nezpracovatelnou zprávu do jiné fronty za účelem dalšího prozkoumání nebo zahození zprávy.

Podrobnosti o zpracování nezpracovatelných zpráv najdete v tématu “Zpracování nezpracovatelných zpráv v produktu IBM MQ classes for JMS” na stránce 203.

Související informace

Určení, že pro běhové prostředí klienta MQI je použit pouze certifikovaný standard FIPS CipherSpecs

Federální standardy zpracování informací (FIPS) pro UNIX, Linux a Windows

Konfigurace prostředků JMS na serveru WebSphere Application Server

Příklady toho, jak definovat vlastnost sharedSubscription

Vlastnost sharedSubscription specifikace aktivace můžete definovat v rámci souboru WebSphere Application Server Liberty `server.xml` . Jinou možností je definovat vlastnost v rámci objektu typu message-driven bean (MDB) pomocí anotací.

Příklad: definování v souboru Liberty `server.xml`

V souboru WebSphere Application Server Liberty `server.xml` definujete specifikaci aktivace, jak je uvedeno v následujícím příkladu. Tento příklad vytvoří trvalý sdílený odběr pro správce front na lokálním hostitel/portu 1490.

```
<jmsActivationSpec id="SubApp/SubscribingEJB/SubscribingMDB" authDataRef="JMSConnectionAlias">
<properties.wmqJms hostName="localhost" port="1490" maxPoolDepth="5"
subscriptionName="MySubName"
subscriptionDurability="DURABLE" sharedSubscription="true"/>
</jmsActivationSpec>
```

Příklad: definování v rámci objektu MDB

Můžete také definovat vlastnost sharedSubscription v rámci objektu MDB pomocí anotací, jak je uvedeno v následujícím příkladu:

```
@ActioncationConfigProperty(propertyName ="sharedSubscription",
propertyValue = "true")
```

Následující příklad ukazuje část kódu MDB, která používá metodu anotací:

```
/**
 * Message-Driven Bean example using Annotations for configuration
```

```
 */
@MessageDriven(
  activationConfig = {
     @ActivationConfigProperty(
        propertyName = "destinationType", propertyValue = "javax.jms.Topic"),
     @ActivationConfigProperty(
        propertyName = "sharedSubscription", propertyValue = "TRUE"),
     @ActivationConfigProperty(
        propertyName = "destination", propertyValue = "JNDI_TOPIC_NAME")
   },
   mappedName = "Stock/IBM")
public class SubscribingMDB implements MessageListener {

   // Default constructor.
   public SubscribingMDB() {
   }

   // @see MessageListener#onMessage(Message)
   public void onMessage(Message message) {
      // implement business logic here
   }

}
```

Související pojmy

“Klonované a sdílené odběry” na stránce 292

V produktu IBM MQ 8.0 nebo novějším existují dvě metody, jak poskytnout více spotřebitelům přístup ke stejnému odběru. Tyto dvě metody jsou buď pomocí klonovaných odběrů, nebo pomocí sdílených odběrů.

Související informace

Odběratelé a odběry

Trvalost odběru

Konfigurace adaptéru prostředků pro odchozí komunikaci

Chcete-li nakonfigurovat odchozí komunikaci, definujte vlastnosti objektu ConnectionFactory a spravovaného cílového objektu.

Příklad použití odchozí komunikace

Při použití odchozí komunikace spouští aplikace spuštěná v aplikačním serveru připojení ke správci front a poté odesílá zprávy do svých front a přijímá zprávy z front synchronním způsobem. Například následující metoda servletu, doGet(), používá odchozí komunikaci:

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
     throws ServletException, IOException {

...

// Look up ConnectionFactory and Queue objects from the JNDI namespace

     InitialContext ic = new InitialContext();
     ConnectionFactory cf = (javax.jms.ConnectionFactory) ic.lookup("myCF");
     Queue q = (javax.jms.Queue) ic.lookup("myQueue");

// Create and start a connection

     Connection c = cf.createConnection();
     c.start();

// Create a session and message producer

     Session s = c.createSession(false, Session.AUTO_ACKNOWLEDGE);
     MessageProducer pr = s.createProducer(q);

// Create and send a message

     Message m = s.createTextMessage("Hello, World!");
     pr.send(m);

// Create a message consumer and receive the message just sent

     MessageConsumer co = s.createConsumer(q);
     Message mr = co.receive(5000);
```

```
// Close the connection
      c.close();
}
```

Když servlet obdrží požadavek GET protokolu HTTP, načte objekt ConnectionFactory a objekt Fronta z oboru názvů JNDI a použije objekty k odeslání zprávy do fronty produktu IBM MQ . Servlet poté přijme zprávu, kterou odeslal.

Prostředky potřebné pro odchozí komunikaci

Chcete-li nakonfigurovat odchozí komunikaci, definujte prostředky Java EE Connector Architecture (JCA) v následujících kategoriích:

- Vlastnosti objektu ConnectionFactory, které aplikační server používá k vytvoření objektu JMS ConnectionFactory .
- Vlastnosti spravovaného objektu místa určení, které aplikační server používá k vytvoření objektu fronty JMS nebo objektu tématu JMS .

Způsob, jakým definujete tyto vlastnosti, závisí na rozhraních pro administraci poskytovaných aplikačním serverem. Objekty ConnectionFactory, Queue a Topic vytvořené aplikačním serverem jsou vázány do oboru názvů JNDI, odkud je lze načíst aplikací.

Zpravidla definujete jeden objekt ConnectionFactory pro každého správce front, ke kterému se mohou aplikace připojit. Definujete jeden objekt fronty pro každou frontu, kterou mohou aplikace potřebovat pro přístup v rámci domény typu point-to-point. A definujete jeden objekt tématu pro každé téma, které aplikace mohou chtít publikovat nebo odebírat. Objekt ConnectionFactory může být nezávislý na doméně. Případně to může být specifické pro doménu, objekt továrny QueueConnectionpro doménu dvoubodového spojení nebo objekt TopicConnectionpro doménu publikování/odběru.

Tip: Pomocí produktu JMS 2.0lze továrnu připojení použít k vytvoření obou připojení a kontextů. Výsledkem je, že je možné mít k dispozici fond připojení přidružený k továrně připojení, která obsahuje kombinaci obou připojení a kontextů. Doporučuje se, aby továrna připojení byla použita pouze pro vytvoření připojení nebo vytvoření kontextů. Tím je zajištěno, že fond připojení pro tuto továrnu připojení bude obsahovat pouze objekty jednoho typu, což bude fond efektivnější.

Vlastnosti objektu ConnectionFactory

Příkaz Tabulka 66 na stránce 432 vypíše vlastnosti objektu ConnectionFactory . Aplikační server použije tyto vlastnosti k vytvoření objektu JMS ConnectionFactory .

Tabulka 66. Vlastnosti objektu ConnectionFactory

Název vlastnosti	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
applicationName	Řetězec	• Název třídy vyvolání, je-li k dispozici, je upraven tak, aby neměl být delší než 28 znaků. Není-li k dispozici, použije se řetězec `WebSphere MQ Client for Java` .	Název, pod kterým je aplikace registrována se správcem front. Tento název aplikace se zobrazí příkazem **DISPLAY CONN MQSC/PCF** (kde se pole nazývá **APPLTAG**) nebo se zobrazí v zobrazení **Připojení aplikace** programu IBM MQ Explorer (kde pole se nazývá **App name**).
brokerCCSubFronta [1]	Řetězec	• **SYSTEM.JMS.ND.CC.SUBSCRIBER.QUEUE** • Název fronty.	Název fronty, z níž spotřebitel připojení přijímá zprávy mimo trvalý odběr.

Tabulka 66. Vlastnosti objektu ConnectionFactory (pokračování)

Název vlastnosti	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
brokerControlFronta [1]	Řetězec	• **SYSTEM.BROKER.CONTROL.QUEUE** • Název fronty.	Název řídicí fronty zprostředkovatele.
brokerPubFronta [1]	Řetězec	• **SYSTEM.BROKER.DEFAULT.STREAM** • Název fronty.	Název fronty, do které jsou odesílány publikované zprávy (fronta proudu).
brokerQueueManager [1]	Řetězec	• **"" (prázdný řetězec)** • Název správce front	Název správce front, v němž je zprostředkovatel spuštěn.
brokerSubFronta [1]	Řetězec	• **SYSTEM.JMS.ND.SUBSCRIBER.QUEUE** • Název fronty.	Název fronty, z níž spotřebitel zpráv netrvalých zpráv přijímá zprávy. Další informace viz vlastnost BROKERSUBQ .
brokerVersion [1]	Řetězec	• **neuvedený** -Po migraci zprostředkovatele z V6 do V7nastavte tuto vlastnost tak, aby záhlaví RFH2 již nebyla použita. Po migraci již tato vlastnost není relevantní. • **V1** -Chcete-li použít zprostředkovatele publikování/odběru IBM MQ . Tato hodnota je výchozí hodnota, je-li hodnota TRANSPORT nastavena na BIND nebo CLIENT. • **V2** -Chcete-li použít zprostředkovatele IBM Integration Bus v nativním režimu. Tato hodnota je výchozí hodnota, je-li hodnota TRANSPORT nastavena na DIRECT nebo DIRECTHTTP.	Verze používaného zprostředkovatele.
ccdtURL	Řetězec	• **null** • Uniform resource locator (adresa URL)	Adresa URL, která identifikuje název a umístění souboru obsahujícího tabulku CCDT (Client Channel Definition CCDT) a určuje, jak lze k souboru přistupovat.
CCSID	Řetězec	• **819** • Identifikátor kódové sady znaků podporovaný virtuálním počítačem Java (JVM)	Identifikátor kódové sady znaků pro připojení.
kanál	Řetězec	• **SYSTEM.DEF.SVRCONN** • Název kanálu MQI	Název kanálu MQI, který má být použit.
cleanupInterval [1]	celé číslo	• **3 600 000** • Kladné celé číslo	Interval, v milisekundách, mezi spuštěními na pozadí obslužného programu čištění publikování/odběru.

Tabulka 66. Vlastnosti objektu ConnectionFactory (pokračování)

Název vlastnosti	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
cleanupLevel [1]	Řetězec	• **Bezpečný** • ŽÁDNÉ • velká • Vynutit • NONDUR	Úroveň vyčištění pro úložiště odběrů založené na zprostředkovateli.
clientID	Řetězec	• **null** • Identifikátor klienta	Identifikátor klienta pro účely připojení.
cloneSupport	Řetězec	• **VYPNUTÉ** -V daném okamžiku může být spuštěna pouze jedna instance trvalého odběratele tématu. • ENABLED-Dvě nebo více instancí stejného odběratele trvalého tématu mohou být spuštěny souběžně, ale každá instance musí být spuštěna v samostatném virtuálním počítači Java (JVM).	Určuje, zda mohou být dvě nebo více instancí stejného odběratele trvalého tématu spuštěny souběžně.
ConnectionNameList	Řetězec	• **localhost (1414)** • Řetězec složený z položek oddělených čárkami, kde každá položka má formát: `HOSTNAME(PORT)` kde *HOSTNAME* je buď název DNS, nebo adresa IP.	Seznam názvů připojení TCP/IP použitých pro odchozí komunikaci. **connectionNameList** nahrazuje vlastnosti **hostname** a **port** . Tato vlastnost se používá k opětovnému připojení ke správcům front s více instancemi. **connectionNameList** je podobný ve tvaru **localAddress**, ale nesmí být zaměňován s ním. Parametr **localAddress** určuje charakteristiky lokální komunikace, zatímco **connectionNameList** určuje, jak se má dostat ke vzdálenému správci front.
FAILIFQUIESCE	Logická hodnota	• **ano** • ne	Pokud se správce front nachází ve stavu uvedení do klidového stavu, nemusí být volání určitých metod neúspěšná.
headerCompression	Řetězec	• **Není** • SYSTEM-Komprese hlavičky zprávy RLE se provádí.	Seznam technik, které lze použít pro kompresi dat záhlaví na připojení.

Tabulka 66. Vlastnosti objektu ConnectionFactory (pokračování)

Název vlastnosti	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
hostName	Řetězec	• **lokální hostitel** • Název hostitele • Adresa IP	Název hostitele nebo adresa IP systému, v němž je správce front umístěn. Vlastnosti **hostname** a **port** jsou nahrazovány vlastností **connectionNameList** , je-li zadána.
localAddress	Řetězec	• **null** • Řetězec ve formátu: `[ host_name ][(low_port [, high_port ])]` kde *název_hostitele* je název hostitele nebo adresa IP, *low_port* a *high_port* jsou čísla portů TCP a hranaté závorky označují volitelnou komponentu.	Pro připojení ke správci front tato vlastnost uvádí jednu nebo obě z následujících možností: • Lokální síťové rozhraní, které má být použito • Lokální port, nebo rozsah lokálních portů, které se mají použít **localAddress** je podobný ve tvaru **connectionNameList**, ale nesmí být zaměňován s ním. Parametr **localAddress** určuje charakteristiky lokální komunikace, zatímco **connectionNameList** určuje, jak se má dostat ke vzdálenému správci front.
messageCompression	Řetězec	• **Není** • Seznam jedné nebo více následujících hodnot oddělených prázdnými znaky: RLE ZLIBFAST ZLIBHIGH	Seznam technik, které lze použít ke komprimování dat zpráv v rámci připojení.
messageSelection [1]	Řetězec	• **client** • BROKER	Určuje, zda je výběr zpráv prováděn produktem IBM MQ classes for JMS nebo zprostředkovatelem. Výběr zpráv zprostředkovatelem není podporován, pokud **brokerVersion** má hodnotu 1.
heslo	Řetězec	• **null** • Heslo	Výchozí heslo, které má být použito při vytváření připojení ke správci front.

Tabulka 66. Vlastnosti objektu ConnectionFactory (pokračování)

Název vlastnosti	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
pollingInterval [1]	celé číslo	• **5000** • Libovolné kladné celé číslo	Pokud každý modul listener pro zprávy v rámci relace nemá ve frontě žádnou odpovídající zprávu, jedná se o maximální interval v milisekundách, který uplyne předtím, než se každý modul listener pro zprávy znovu pokusí získat zprávu z příslušné fronty. Pokud se často stává, že pro žádný z listenerů zpráv v rámci relace není k dispozici žádná vhodná zpráva, zvažte zvýšení hodnoty této vlastnosti. Tato vlastnost je relevantní pouze v případě, že **TRANSPORT** má hodnotu `BIND` nebo `CLIENT`.
Port	celé číslo	• **1414** • Číslo portu TCP	Port, na kterém správce front naslouchá. Vlastnosti **hostname** a **port** jsou nahrazovány vlastností **connectionNameList** , je-li zadána.
providerVersion	řetězec	• **nespecifikováno** • Řetězec v jednom z následujících formátů – V.R.M.F – V.R.M – V.R – V kde V, R, M a F jsou celočíselné hodnoty větší než nebo rovné nule.	Verze, vydání, úroveň modifikace a opravný balík správce front, ke kterému se tato aplikace hodlá připojovat.
Interval pubAck [1]	celé číslo	• **25** • Kladné celé číslo	Počet zpráv publikovaných vydavatelem, než produkt IBM MQ classes for JMS požádá o potvrzení od zprostředkovatele.
queueManager	Řetězec	• **"" (prázdný řetězec)** • Název správce front	Název správce front, s nímž má být navázáno připojení.
receiveExit [3]	Řetězec	• **null** • Řetězec obsahující jednu nebo více položek oddělených čárkami, kde každá položka představuje úplný název třídy, která implementuje rozhraní produktu IBM MQ classes for Java , MQReceiveExit .	Identifikuje uživatelský program pro příjem kanálu nebo posloupnost ukončovacích ukončovacích programů, které mají být spuštěny v posloupnosti.

Tabulka 66. Vlastnosti objektu ConnectionFactory (pokračování)

Název vlastnosti	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
receiveExitInit	Řetězec	• **null** • Řetězec obsahující jednu nebo více položek dat uživatele oddělených čárkami	Uživatelská data, která jsou předána uživatelským programům přijetí kanálu při jejich vyvolání.
rescanInterval [1]	celé číslo	• **5000** • Libovolné kladné celé číslo	Pokud spotřebitel zpráv v doméně typu point-to-point používá selektor zpráv k výběru zpráv, které chce přijímat, produkt IBM MQ classes for JMS prohledá frontu IBM MQ pro vhodné zprávy v posloupnosti určené atributem **MsgDeliverySequence** fronty. Pokud produkt IBM MQ classes for JMS najde vhodnou zprávu a doručí ji spotřebiteli, produkt IBM MQ classes for JMS obnoví hledání další vhodné zprávy z aktuální pozice ve frontě. Produkt IBM MQ classes for JMS pokračuje v hledání fronty tímto způsobem, dokud nedosáhne konce fronty, nebo dokud nevyprší časový interval v milisekundách určený hodnotou této vlastnosti. V každém případě se IBM MQ classes for JMS vrátí na začátek fronty, aby pokračoval v hledání, a začne nový časový interval.
securityExit [3]	Řetězec	• **null** • Úplný název třídy, která implementuje rozhraní IBM MQ classes for Java , MQSecurityExit .	Identifikuje uživatelský program zabezpečení kanálu.
Init securityExit	Řetězec	• **null** • Řetězec uživatelských dat	Uživatelská data, která jsou předána do uživatelského programu zabezpečení kanálu, je-li volán.
SENDCHECKCOUNT	celé číslo	• **0** • Libovolné kladné celé číslo	Počet volání odesílání, která umožňují mezi kontrolou asynchronních chyb vložení, v rámci jedné relace JMS bez transakce.
sendExit [3]	Řetězec	• **null** • Řetězec obsahující jednu nebo více položek oddělených čárkami, kde každá položka představuje úplný název třídy, která implementuje rozhraní IBM MQ classes for Java , MQSendExit .	Identifikuje uživatelský program odeslání kanálu nebo posloupnost ukončovacích ukončovacích programů, které mají být spuštěny za sebou.

Tabulka 66. Vlastnosti objektu ConnectionFactory (pokračování)			
Název vlastnosti	**Typ**	**Platné hodnoty (výchozí hodnota tučně)**	**Popis**
SENDEXITINIT	Řetězec	• **null** • Řetězec obsahující jednu nebo více položek dat uživatele oddělených čárkami	Data uživatele, která jsou předána uživatelským programům odeslání kanálu, když jsou volány.
SHARECONVALLOWED	Logická hodnota	• **NO** -Připojení klienta nemůže sdílet svůj soket. • YES-Připojení klienta může sdílet svůj soket.	Určuje, zda může připojení klienta sdílet svůj soket s dalšími připojeními JMS nejvyšší úrovně ze stejného procesu ke stejnému správci front, pokud se definice kanálu shodují.
sparseSubscriptions [1]	Logická hodnota	• **false** -Odběry přijímají časté odpovídající zprávy. • true-Odběry přijímají odpovídající zprávy zřídka. Tato hodnota vyžaduje, aby byla fronta odběru otevřena pro procházení.	Řídí zásadu načítání zpráv objektu TopicSubscriber .
Úložiště sslCert	Řetězec	• **null** • Řetězec jedné nebo více adres URL LDAP oddělených mezerami. Každá adresa URL protokolu LDAP má tento formát: `ldap://host_name [: port ]` kde *název_hostitele* je název hostitele nebo adresa IP, *port* je číslo portu TCP a hranaté závorky označují volitelnou komponentu.	Servery LDAP (Lightweight Directory Access Protocol), které zadržují seznamy zrušených certifikátů (CRL) pro použití v připojení TLS.
SSLCIPHERSUITE	Řetězec	• **null** • Název sady CipherSuite	Sada CipherSuite , která má být použita pro připojení TLS.
sslFipsPovinné [2]	Logická hodnota	• **ne** • ano	Určuje, zda připojení TLS musí používat sadu CipherSuite podporovanou poskytovatelem FIPS JSSE FIPS produktu IBM Java (IBMJSSEFIPS).
SSLPEERNAME	Řetězec	• **null** • Šablona pro rozlišující názvy	Pro připojení TLS se jedná o šablonu, která se používá ke kontrole rozlišujícího názvu v digitálním certifikátu poskytovaného správcem front.
SSLRESETCOUNT	celé číslo	• **0** • Celé číslo v rozsahu 0 až 999 999 999.	Celkový počet bajtů odeslaných a přijatých připojením TLS dříve, než jsou znovu vyjednány tajné klíče použité TLS.

Tabulka 66. Vlastnosti objektu ConnectionFactory (pokračování)

Název vlastnosti	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
Továrna sslSocket	Řetězec	Řetězec představující plně kvalifikovaný název třídy třídy poskytující implementaci rozhraní javax.net.ssl.SSLSocketFactory , volitelně včetně argumentu, který má být předán metodě konstruktoru, uzavřený v závorkách.	Veškerá připojení vytvořená v oboru administrovaného cílového objektu používají sokety získané z této implementace rozhraní SSLSocketFactory .
statusRefreshInterval [1]	celé číslo	• **60000** • Libovolné kladné celé číslo	Interval (v milisekundách) mezi aktualizacemi transakce s dlouhou dobou zpracování, která zjišťuje, zda odběratel ztratil připojení ke správci front. Tato vlastnost je relevantní pouze v případě, že má **SUBSTORE** hodnotu QUEUE.
subscriptionStore [1]	Řetězec	• **BROKER** • MIGRATE • QUEUE	Určuje, kam IBM MQ classes for JMS ukládá trvalá data o aktivních odběrech.
Odpovídající targetClient	Logická hodnota	• **ano** • ne	Určuje, zda zpráva s odpovědí odeslaná do fronty určené polem záhlaví JMSReplyTo příchozí zprávy má záhlaví MQRFH2 pouze v případě, že má příchozí zpráva záhlaví MQRFH2 .

Tabulka 66. Vlastnosti objektu ConnectionFactory (pokračování)

Název vlastnosti	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
temporaryModel	Řetězec	• **SYSTEM.DEFAULT.MODEL.QUEUE** • SYSTEM.JMS.TEMPQ.MODEL • Libovolný řetězec	Název modelové fronty, ze které jsou vytvářeny dočasné fronty produktu JMS . Použijte `SYSTEM.DEFAULT.MODEL.QUEUE` , pokud jsou obě následující příkazy pravdivé: • Vaše aplikace používá dočasnou frontu, která bude přijímat přechodné zprávy. • Pouze jedna aplikace vytvoří dočasnou frontu ve správci front, na který odkazuje ConnectionFactory , na určitou dobu. Všimněte si, že `SYSTEM.DEFAULT.MODEL.QUEUE` lze v daném okamžiku otevřít pouze jednou aplikací. Použijte `SYSTEM.JMS.TEMPQ.MODEL` v následujících situacích: • Když vaše aplikace používá dočasnou frontu, která bude přijímat trvalé zprávy. • Pokud se může více aplikací připojit ke správci front, který ukazuje ConnectionFactory , a tyto aplikace potřebují vytvořit dočasné fronty současně. Definujte novou modelovou frontu s atributem **DEFPSIST** nastaveným na hodnotu YESa atribut **DEFSOPT** se nastaví na hodnotu `SHARED` v této situaci: • Pokud vaše aplikace používá dočasnou frontu, která bude přijímat přechodné zprávy, a více aplikací se připojí ke správci front, na který odkazuje ConnectionFactory , a tyto aplikace budou muset vytvořit dočasné fronty současně. Když se vytvoří nová modelová fronta, nastavte vlastnost **`temporaryModel`** na název nové modelové fronty.

Tabulka 66. Vlastnosti objektu ConnectionFactory (pokračování)

Název vlastnosti	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
tempQPrefix	Řetězec	• **"" (prázdný řetězec)** • Předpona, která může být použita k vytvoření názvu dynamické fronty IBM MQ . Pravidla pro vytváření předpony jsou stejná jako pravidla pro vytváření obsahu pole **DynamicQName** v deskriptoru objektu IBM MQ , struktura MQOD, ale poslední nemezerový znak musí být hvězdička (*). Je-li hodnotou vlastnosti prázdný řetězec, produkt IBM MQ classes for JMS použije hodnotu AMQ.* při vytváření dynamické fronty.	Předpona, která se používá k vytvoření názvu dynamické fronty produktu IBM MQ .
TEMPTOPICPREFIX	Řetězec	Jakýkoli řetězec, který není null, obsahuje pouze platné znaky pro řetězec tématu IBM MQ .	Při vytváření dočasných témat produkt JMS vygeneruje řetězec tématu ve tvaru "TEMP/*TEMPTOPICPREFIX*/*jedinečné_id* ", nebo je-li tato vlastnost ponechána s výchozí hodnotou, pouze "TEMP/*jedinečné_id*". Zadáte-li neprázdný **TEMPTOPICPREFIX** , umožníte definovat specifické modelové fronty pro vytvoření spravovaných front pro odběratele na dočasná témata vytvořená pod tímto připojením.

Tabulka 66. Vlastnosti objektu ConnectionFactory (pokračování)			
Název vlastnosti	**Typ**	**Platné hodnoty (výchozí hodnota tučně)**	**Popis**
transportType	Řetězec	• **client** • Vazby • BINDINGS_THEN_CLIENT	Určuje, zda připojení ke správci front používá režim klienta nebo režim vazeb. Je-li zadána hodnota `BINDINGS_THEN_CLIENT` , adaptér prostředků se nejprve pokusí vytvořit připojení v režimu vazeb. Dojde-li k selhání tohoto pokusu o připojení, adaptér prostředků se poté pokusí o připojení v režimu klienta. z/OS Pokud byla specifikace aktivace spuštěná na systému WebSphere Application Server for z/OS konfigurována tak, aby používala režim přenosu `BINDINGS_THEN_CLIENT` a dříve zavedené připojení je přerušeno, provede se pokus o opětovné připojení ze specifikace aktivace při prvním pokusu o použití režimu přenosu `BINDINGS` . Pokud se pokus o připojení režimu přenosu `BINDINGS` nezdaří, specifikace aktivace se následně pokusí o připojení v režimu transportu `CLIENT` .
jméno uživatele	Řetězec	• **null** • Jméno uživatele	Výchozí jméno uživatele, které má být použito při vytváření připojení ke správci front.
wildcardFormat	celé číslo	• CHAR-Rozlišuje se pouze znakové zástupné znaky, jak je použito ve zprostředkovateli verze 1 • TOPIC-Rozpoznává pouze zástupné znaky na úrovni témat, jak je použito ve zprostředkovateli verze 2	Která verze syntaxe zástupných znaků se má použít.

Notes:

1. Tuto vlastnost lze použít s produktem IBM WebSphere MQ classes for JMS v produktu IBM WebSphere MQ 7.0 , ale nemá vliv na aplikaci připojenou ke správci front produktu IBM WebSphere MQ 7.0 , pokud není vlastnost providerVersion nastavena na číslo verze nižší než 7.
2. Důležité informace o použití vlastnosti sslFipsRequired naleznete v příručce “Omezení adaptéru prostředků produktu IBM MQ” na stránce 404.
3. Informace o tom, jak nakonfigurovat adaptér prostředků tak, aby mohl najít uživatelskou proceduru, najdete v tématu “Konfigurace produktu IBM MQ classes for JMS pro použití uživatelských procedur kanálu” na stránce 250.

Následující příklad ukazuje typickou sadu vlastností objektu ConnectionFactory :

```
channel:        SYSTEM.DEF.SVRCONN
```

```
hostName:       192.168.0.42
port:           1414
queueManager:   ExampleQM
transportType:  CLIENT
```

Vlastnosti spravovaného objektu cíle

Aplikační server používá vlastnosti spravovaného cílového objektu k vytvoření objektu fronty JMS nebo objektu tématu JMS .

Příkaz Tabulka 67 na stránce 443 uvádí seznam vlastností, které jsou společné pro objekt fronty a objekt tématu.

Tabulka 67. Vlastnosti, které jsou společné pro objekt fronty a objekt Topic

Název vlastnosti	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
CCSID	Řetězec	• **1208** • Identifikátor kódové sady znaků podporovaný virtuálním počítačem Java (JVM)	Identifikátor kódové sady znaků pro místo určení.
kódování	Řetězec	• **Nativní** • Řetězec tří znaků: – První znak určuje reprezentaci binárních celých čísel: - *N* označuje normální kódování. - *R* označuje inverzní kódování. – Druhý znak určuje reprezentaci pakovaných dekadických celých čísel: - *N* označuje normální kódování. - *R* označuje inverzní kódování. – Třetí znak určuje reprezentaci čísel s pohyblivou řádovou čárkou: - *N* označuje standardní kódování IEEE. - *R* označuje inverzní kódování IEEE. - *3* označuje kódování zSeries . NATIVE je ekvivalentní řetězci NNN.	Znázornění binárních celých čísel, pakovaných desetinných čísel a čísel s pohyblivou řádovou čárkou pro místo určení.
Vypršení	Řetězec	• **APP** -Doba vypršení platnosti zprávy je určena producentem zpráv. • UNLIM-Platnost zprávy nikdy nevyprší. • 0-Platnost zprávy nikdy nevyprší. • Kladné celé číslo představující dobu vypršení platnosti zprávy v milisekundách.	Doba vypršení platnosti zprávy odeslané do místa určení.
FAILIFQUIESCE	Řetězec	• **ano** • ne	Zda se pokus o přístup k místu určení nezdaří, je-li správce front ve stavu uvedení do klidového stavu.

Tabulka 67. Vlastnosti, které jsou společné pro objekt fronty a objekt Topic (pokračování)

Název vlastnosti	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
Styl messageBody	Řetězec	• **Neuvedeno** • JMS • MQ	Vlastnost **messageBodyStyle** lze nastavit ve frontách a tématech produktu JMS : UNSPECIFIED (výchozí) • Při odesílání produkt IBM MQ classes for JMS generuje a zahrnuje záhlaví MQRFH2 v závislosti na hodnotě WMQ_TARGET_CLIENT. • Při příjmu produkt IBM MQ classes for JMS nastaví vlastnosti zprávy produktu JMS podle hodnot v záhlaví MQRFH2, je-li přítomno. MQRFH2 není prezentován jako část těla zprávy JMS . JMS • Při odesílání produkt IBM MQ classes for JMS automaticky vygeneruje záhlaví MQRFH2 a zahrne záhlaví do zprávy produktu IBM MQ . • Při příjmu produkt IBM MQ classes for JMS nastaví vlastnosti zprávy produktu JMS podle hodnot v záhlaví MQRFH2, je-li přítomno. MQRFH2 není prezentován jako část těla zprávy JMS . MQ • Při odesílání produkt IBM MQ classes for JMS negeneruje MQRFH2. • Když přijímá, IBM MQ classes for JMS prezentuje MQRFH2 jako část těla zprávy JMS .

Tabulka 67. Vlastnosti, které jsou společné pro objekt fronty a objekt Topic (pokračování)

Název vlastnosti	Typ	Platné hodnoty (výchozí hodnota tučně)	Popis
trvání, perzistence	Řetězec	• **APP** -Trvalost zprávy je určena producentem zpráv. • QDEF-Perzistence zprávy je určena atributem **DefPersistence** fronty IBM MQ . • PERS-Zpráva je trvalá. • NON-Zpráva je přechodná. • HIGH-Trvalost zprávy je určena atributem **NonPersistentMessageClass** fronty IBM MQ podle vysvětlení v "Trvalé zprávy produktu JMS" na stránce 218.	Perzistence zprávy odeslané do místa určení.
priority (priorita)	Řetězec	• **APP** -Priorita zprávy je určena producentem zpráv. • QDEF-Priorita zprávy je určena atributem **DefPriority** fronty IBM MQ . • Celé číslo v rozsahu 0, nejnižší priorita, až 9, nejvyšší priorita.	Priorita zprávy odeslané do místa určení.
PUTASYNCALLOWED	Řetězec	• QUEUE-Určení, zda jsou asynchronní operace vložení povoleny odkazem na definici fronty. • TOPIC-Určete, zda jsou asynchronní operace vložení povoleny odkazem na definici tématu. • DESTINATION-Určení, zda asynchronní operace vložení jsou povoleny odkazem na definici fronty nebo tématu. • DISABLED-Asynchronní operace put nejsou povoleny. • ENABLED-Asynchronní vkládání jsou povoleny.	Určuje, zda mají producenti zpráv povoleno používat asynchronní operace put k odesílání zpráv do tohoto místa určení.
READAHEADALLOWED	celé číslo	• **DESTINATION** -Určení toho, zda je dopředné čtení povoleno odkazem na definici fronty nebo tématu. • DISABLED-Čtení napřed není povoleno. • ENABLED-Čtení napřed je povoleno. • QUEUE-Určení, zda je dopředné čtení povoleno s odkazem na definici fronty. • TOPIC-Určení, zda je dopředné čtení povoleno s odkazem na definici tématu.	Určuje, zda mají spotřebitelé zpráv a ve frontě zpráv povoleno používat dopředné čtení k získání přechodných zpráv z místa určení do interní vyrovnávací paměti, než je přijme.

Tabulka 67. Vlastnosti, které jsou společné pro objekt fronty a objekt Topic (pokračování)			
Název vlastnosti	**Typ**	**Platné hodnoty (výchozí hodnota tučně)**	**Popis**
receiveCCSID	celé číslo	• **0** -použít prostředí JVM `Charset.defaultCharset` • 1208- UTF-8 • Podporovaný identifikátor kódované znakové sady	Vlastnost místa určení, která nastavuje cílové CCSID pro převod zpráv správce front. Hodnota je ignorována, pokud parametr **`receiveConversion`** není nastaven na hodnotu `QMGR`.
receiveConversion	Řetězec	• **CLIENT_MSG** • QMGR	Cílová vlastnost, která určuje, zda bude převod dat prováděn správcem front.
targetClient	Řetězec	• **JMS** -Cílem zprávy je aplikace JMS . • MQ -Cílem zprávy je aplikace jiného typu než produktuJMS IBM MQ .	Určuje, zda má být cílem zprávy odeslané do cíle aplikace JMS . Zpráva s cílem, který je aplikací produktu JMS , obsahuje záhlaví MQRFH2 .

Produkt Tabulka 68 na stránce 446 obsahuje seznam vlastností, které jsou specifické pro objekt Queue.

Tabulka 68. Vlastnosti, které jsou specifické pro objekt fronty			
Název vlastnosti	**Typ**	**Platné hodnoty (výchozí hodnota tučně)**	**Popis**
baseQueueManagerName	Řetězec	• **"" (prázdný řetězec)** • Název správce front	Název správce front, který vlastní základní frontu produktu IBM MQ .
Název baseQueue	Řetězec	• **"" (prázdný řetězec)** • Název fronty.	Název základní fronty produktu IBM MQ .

Produkt Tabulka 69 na stránce 446 obsahuje seznam vlastností, které jsou specifické pro objekt tématu.

Tabulka 69. Vlastnosti, které jsou specifické pro objekt tématu			
Název vlastnosti	**Typ**	**Platné hodnoty (výchozí hodnota tučně)**	**Popis**
Název baseTopic	Řetězec	• **"" (prázdný řetězec)** • Název tématu	Název základního tématu.
brokerCCDurSubQueue [1]	Řetězec	• **SYSTEM.JMS.D.CC.SUBSCRIBER.QUEUE** • Název fronty.	Název fronty, z níž spotřebitel připojení přijímá zprávy trvalého odběru.
brokerDurSubQueue [1]	Řetězec	• **SYSTEM.JMS.D.SUBSCRIBER.QUEUE** • Název fronty.	Název fronty, ze které odběratel trvalého tématu přijímá zprávy. Další informace viz vlastnost **BROKEDURRSUBQ** v dokumentaci produktu IBM MQ Explorer .

Tabulka 69. Vlastnosti, které jsou specifické pro objekt tématu (pokračování)			
Název vlastnosti	**Typ**	**Platné hodnoty (výchozí hodnota tučně)**	**Popis**
brokerPubFronta [1]	Řetězec	• **Nenastaveno** • Název fronty.	Název fronty, do které jsou odesílány publikované zprávy (fronta proudu). Hodnota této vlastnosti přepíše hodnotu vlastnosti **brokerPubQueue** objektu ConnectionFactory . Pokud však nenastavíte hodnotu této vlastnosti, použije se místo toho hodnota vlastnosti **brokerPubQueue** objektu ConnectionFactory .
brokerPubQueueManager [1]	Řetězec	• **"" (prázdný řetězec)** • Název správce front	Název správce front, který vlastní frontu, do níž jsou odesílány zprávy publikované v rámci daného tématu.
brokerVersion [1]	Řetězec	• **Nenastaveno** • 1 • 2	Verze používaného zprostředkovatele. Hodnota této vlastnosti přepíše hodnotu vlastnosti **brokerVersion** objektu ConnectionFactory . Pokud však nenastavíte hodnotu této vlastnosti, použije se místo toho hodnota vlastnosti **brokerVersion** objektu ConnectionFactory .

Poznámka:

1. Tuto vlastnost lze použít s produktem IBM WebSphere MQ classes for JMS v produktu IBM WebSphere MQ 7.0 , ale neovlivňuje aplikaci připojenou ke správci front produktu IBM WebSphere MQ 7.0 , pokud není vlastnost providerVersion objektu ConnectionFactory nastavena na číslo verze menší než 7.

Následující příklad ukazuje sadu vlastností objektu Queue:

```
expiry:               UNLIM
persistence:          QDEF
baseQueueManagerName: ExampleQM
baseQueueName:        SYSTEM.JMS.TEMPQ.MODEL
```

Následující příklad zobrazuje sadu vlastností objektu Topic:

```
expiry:               UNLIM
persistence:          NON
baseTopicName:        myTestTopic
```

Související informace

Určení, že pro běhové prostředí klienta MQI je použit pouze certifikovaný standard FIPS CipherSpecs
Federální standardy zpracování informací (FIPS) pro UNIX, Linux, and Windows
Konfigurace prostředků JMS na serveru WebSphere Application Server

V 9.0.0.5 *Konfigurace vlastnosti Vyhovující klienta targetClientpro specifikaci aktivace*

Můžete nakonfigurovat vlastnost **targetClientMatching** pro specifikaci aktivace tak, aby bylo záhlaví MQRFH2 zahrnuto ve zprávách odpovědi, když zprávy požadavku neobsahují záhlaví MQRFH2 . To znamená, že všechny vlastnosti zprávy, které aplikace definuje na zprávě s odpovědí, jsou zahrnuty při odeslání zprávy.

Informace o této úloze

Pokud aplikace objektu typu message-driven bean (MDB) spotřebovává zprávy, které neobsahují záhlaví MQRFH2 , prostřednictvím specifikace aktivace adaptéru prostředků JCA IBM MQ a následně odesílá zprávy odpovědi na místo určení rozhraní JMS vytvořené z pole JMSReplyTo zprávy požadavku, zprávy odpovědi musí obsahovat záhlaví MQRFH2 , i když zprávy požadavku nepracují, jinak budou ztraceny všechny vlastnosti zprávy, které aplikace definovala ve zprávě odpovědi.

Vlastnost **targetClientMatching** definuje, zda zpráva odpovědi odeslaná do fronty označené v poli záhlaví JMSReplyTo příchozí zprávy má záhlaví MQRFH2 pouze v případě, že má příchozí zpráva záhlaví MQRFH2 . Tuto vlastnost je možné nakonfigurovat pro specifikaci aktivace v produktu WebSphere Application Server traditional i v produktu WebSphere Application Server Liberty.

Nastavíte-li hodnotu vlastnosti **targetClientMatching** na hodnotu `false`, záhlaví MQRFH2 lze zahrnout do zprávy odpovědi odeslané do cíle rozhraní JMS vytvořeného z záhlaví JMSReplyTo příchozí zprávy požadavku, která neobsahuje MQRFH2. Důvodem je skutečnost, že vlastnost **targetClient** na cíli JMS je nastavena na hodnotu `0`, což znamená, že zprávy obsahují záhlaví MQRFH2 . Přítomnost záhlaví MQRFH2 v odchozí zprávě umožňuje ukládání vlastností zpráv definovaných uživatelem ve zprávě při odeslání do fronty produktu IBM MQ .

Je-li vlastnost **targetClientMatching** nastavena na hodnotu `true` a zpráva požadavku neobsahuje záhlaví MQRFH2 , záhlaví MQRFH2 nebude zahrnuto do zprávy odpovědi.

Procedura

- V produktu WebSphere Application Server traditionalpomocí administrativní konzoly definujte vlastnost produktu **targetClientMatching** jako přizpůsobenou vlastnost ve specifikaci aktivace produktu IBM MQ :
  a) V navigačním podokně klepněte na **Prostředky-> JMS-> Specifikace aktivace**.
  b) Vyberte název specifikace aktivace, kterou chcete zobrazit nebo změnit.
  c) Klepněte na volbu **Přizpůsobené vlastnosti-> Nový** a poté zadejte podrobnosti nové přizpůsobené vlastnosti.
  Nastavte název vlastnosti na `targetClientMatching`, typ na `java.lang.Boolean` a hodnotu na `false`.
- V WebSphere Application Server Libertyzadejte vlastnost **targetClientMatching** na definici specifikace aktivace v rámci `server.xml`.
  Příklad:

```
<jmsActivationSpec id="SimpleMDBApplication/SimpleEchoMDB/SimpleEchoMDB">
<properties.wmqJms destinationRef="MDBRequestQ"
queueManager="MY_QMGR" transportType="BINDINGS" targetClientMatching="false"/>
<authData password="********" user="tom"/>
</jmsActivationSpec>
```

Související pojmy

“Vytvoření cílů v aplikaci JMS” na stránce 192
Místo načítání cílů jako spravovaných objektů z oboru názvů rozhraní JNDI (Java Naming and Directory Interface) může aplikace produktu JMS použít relaci k dynamickému vytváření cílů v době běhu programu. Aplikace může použít identifikátor URI (Uniform Resource Identifier) k identifikaci fronty produktu IBM MQ nebo tématu a volitelně také k určení jedné či více vlastností objektu Fronta nebo Téma.

“Konfigurace adaptéru prostředků pro odchozí komunikaci” na stránce 431

Chcete-li nakonfigurovat odchozí komunikaci, definujte vlastnosti objektu ConnectionFactory a spravovaného cílového objektu.

Ověření instalace adaptéru prostředků

Program IVT (installation verification test) pro adaptér prostředků produktu IBM MQ je dodáván jako soubor EAR. Chcete-li použít tento program, musíte jej implementovat a definovat některé objekty jako prostředky produktu JCA .

Informace o této úloze

Program pro test verifikace instalace (IVT) je dodáván jako soubor podnikového archivu (EAR) s názvem `wmq.jmsra.ivt.ear`. Tento soubor je nainstalován s produktem IBM MQ classes for JMS ve stejném adresáři jako soubor RAR adaptéru prostředků produktu IBM MQ , `wmq.jmsra.rar`. Informace o tom, kde jsou tyto soubory nainstalovány, najdete v tématu “Instalace adaptéru prostředků produktu IBM MQ” na stránce 407.

Na svém aplikačním serveru musíte implementovat program IVT. Program IVT obsahuje servlet a objekt MDB, který testuje, zda lze zprávu odesílat a přijímat z fronty produktu IBM MQ . Program IVT můžete použít k ověření, že adaptér prostředků IBM MQ byl správně nakonfigurován pro podporu distribuovaných transakcí. Pokud implementujete adaptér prostředků produktu IBM MQ v jiném aplikačním serveru nežIBM , služba IBM vás může požádat, abyste demonstrovali, že IVT pracuje pro ověření, že je váš aplikační server správně nakonfigurovaný.

Než budete moci spustit program IVT, musíte definovat objekt ConnectionFactory , objekt fronty a případně objekt specifikace aktivace jako prostředky produktu JCA a zajistit, aby váš aplikační server vytvořil objekty produktu JMS z těchto definic a svázal je s oborem názvů JNDI . Můžete zvolit vlastnosti objektů, které se budou shodovat s nastavením hostitele a portu vlastního objektu QueueManager, ale následující sadou vlastností je jednoduchý příklad:

```
ConnectionFactory object:
channel:              SYSTEM.DEF.SVRCONN
hostName:             localhost
port:                 1550
queueManager:         QM1
transportType:        CLIENT
Queue object:
baseQueueManagerName: QM1
baseQueueName:        TEST.QUEUE
```

Mechanismus použitý k definování objektů specifikace ConnectionFactory, Queue a Activation Specification se liší v závislosti na aplikačním serveru. Chcete-li například nastavit tyto vlastnosti v rámci produktu WebSphere Application Server Liberty, přidejte do souboru `server.xml` aplikačního serveru následující položky:

```
<!-- IVT Connection factory -->
<jmsQueueConnectionFactory connectionManagerRef="ConMgrIVT" jndiName="IVTCF">
   <properties.wmqJms channel="SYSTEM.DEF.SVRCONN" hostname="localhost" port="1550"
transportType="CLIENT"/>
</jmsQueueConnectionFactory>
<connectionManager id="ConMgrIVT" maxPoolSize="10"/>

<!-- IVT Queues -->
<jmsQueue id="IVTQueue" jndiName="IVTQueue">
   <properties.wmqJms baseQueueName="TEST.QUEUE"/>
</jmsQueue>

<!-- IVT Activation Spec -->
<jmsActivationSpec id="wmq.jmsra.ivt/WMQ_IVT_MDB/WMQ_IVT_MDB">
   <properties.wmqJms destinationRef="IVTQueue"
transportType="CLIENT"
queueManager="QM1"
hostName="localhost"
port="1550"
maxPoolDepth="1"/>
</jmsActivationSpec>
```

Program IVT standardně očekává, že objekt ConnectionFactory má být svázán v oboru názvů JNDI s názvem jms/ivt/IVTCF a s objektem fronty, který má být svázán s názvem jms/ivt/IVTQueue. Můžete použít různé názvy, ale pokud ano, musíte zadat názvy objektů na počáteční stránce programu IVT a upravit odpovídajícím způsobem soubor EAR.

Po implementaci programu IVT a aplikačního serveru vytvořili objekty produktu JMS a svázali je s oborem názvů JNDI , můžete spustit program IVT provedením následujících kroků.

Postup

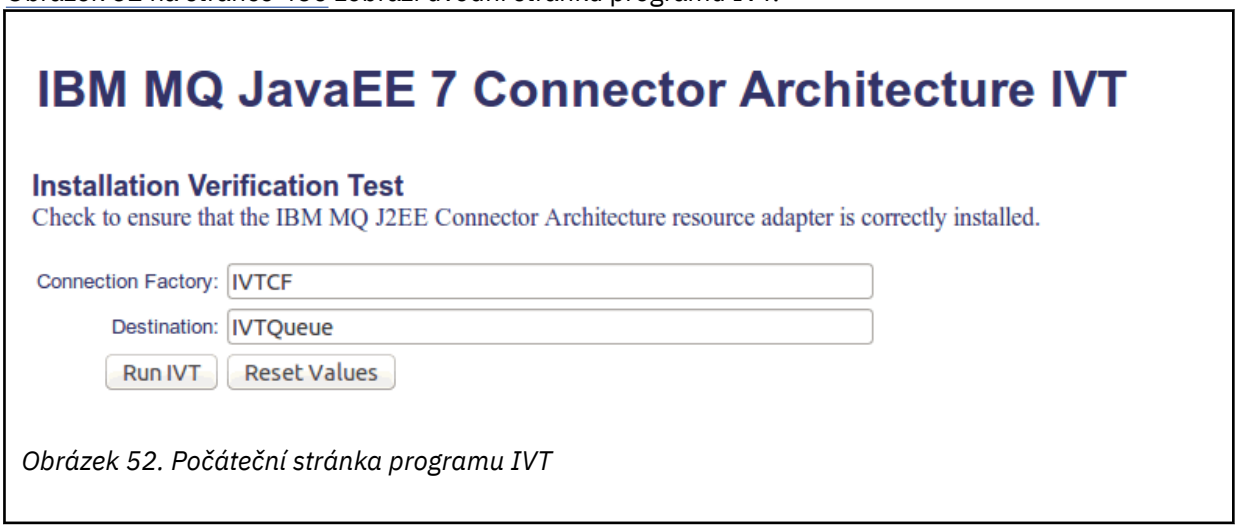
1. Spusťte program IVT zadáním adresy URL v následujícím formátu do svého webového prohlížeče:

   ```
   http://app_server_host: port/WMQ_IVT/
   ```

   kde *hostitel_aplikačního_serveru* je adresa IP nebo název hostitele systému, na kterém běží váš aplikační server, a *port* je číslo portu TCP, na kterém naslouchá aplikační server. Příklad:

   ```
   http://localhost:9080/WMQ_IVT/
   ```

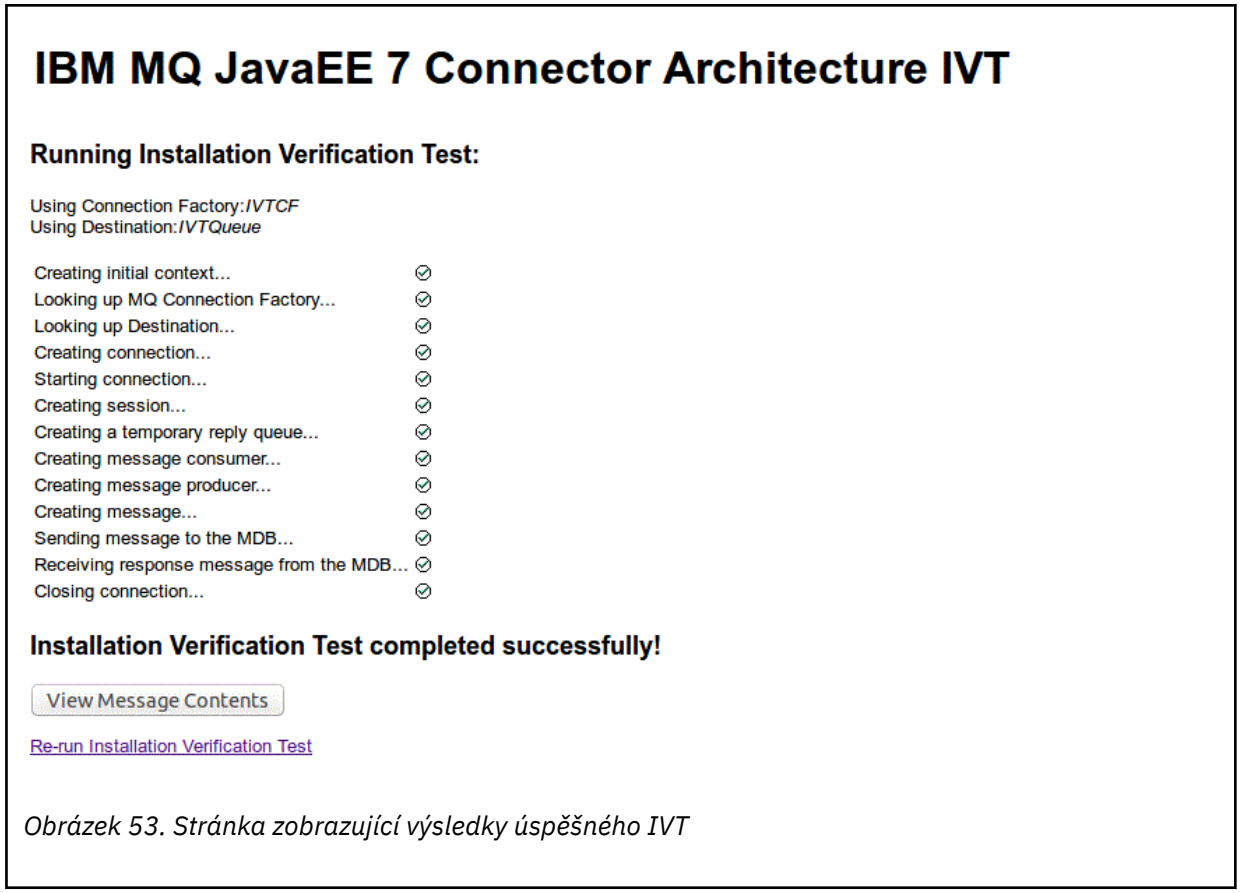
   Obrázek 52 na stránce 450 zobrazí úvodní stránku programu IVT.

   Obrázek 52. Počáteční stránka programu IVT

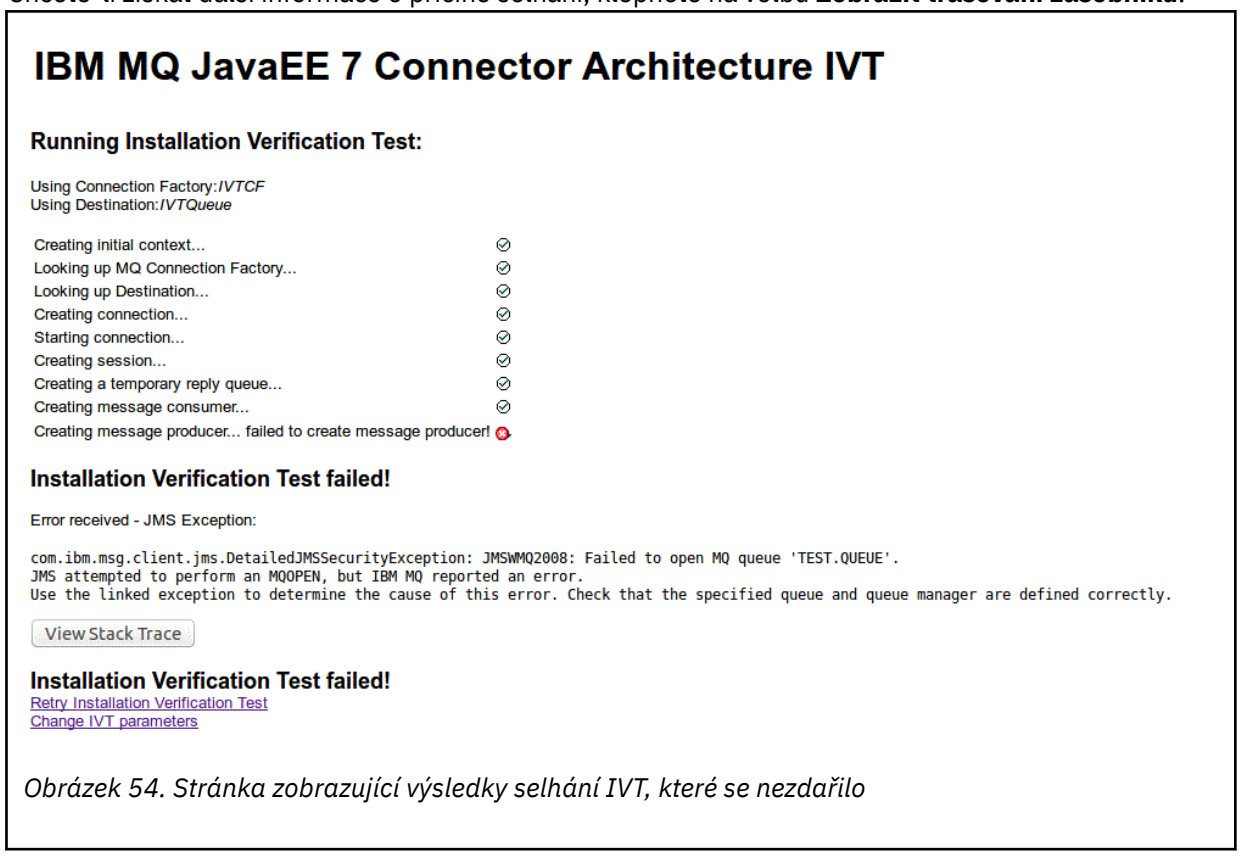
2. Chcete-li spustit test, klepněte na volbu **Spustit IVT**.

   Obrázek 53 na stránce 451 zobrazí stránku, která se zobrazí, je-li IVT úspěšný.



Obrázek 53. Stránka zobrazující výsledky úspěšného IVT

Pokud IVT selže, zobrazí se stránka podobná té, která je zobrazena v Obrázek 54 na stránce 451 . Chcete-li získat další informace o příčině selhání, klepněte na volbu **Zobrazit trasování zásobníku**.



Obrázek 54. Stránka zobrazující výsledky selhání IVT, které se nezdařilo

Windows Instalace a testování adaptéru prostředků v serveru GlassFish Server

Chcete-li instalovat adaptér prostředků produktu IBM MQ na server GlassFish Server v operačním systému Windows , musíte nejprve vytvořit a spustit doménu. Poté můžete implementovat a konfigurovat adaptér prostředků a implementovat a spustit aplikaci testu verifikace instalace (IVT).

Informace o této úloze

Důležité: Tyto pokyny jsou určeny pro produkt GlassFish Server verze 4.

Tato úloha předpokládá, že máte spuštěný aplikační server GlassFish Server, a že jste obeznámeni se standardními administračními úlohami pro tuto úlohu. Tato úloha také předpokládá, že máte instalaci produktu IBM MQ na svém lokálním systému a že jste obeznámeni se standardními úlohami administrace.

Poznámka: Chcete-li dokončit následující kroky úlohy, musíte mít funkční instalaci produktu IBM MQ s nakonfigurovanými následujícími objekty:

- Správce front s názvem QM, který je spuštěn na portu 1414, který používá kanál SYSTEM.DEF.SVRCONN, který se připojuje pomocí přenosu klienta.
- Fronta s názvem Q1.

Postup

1. Spusťte program shellu serveru GlassFish Server **asadmin** .
   a) Otevřete příkazový řádek Windows a přejděte do adresáře *GlassFish*/bin , kde *GlassFish* je adresář, kde je nainstalován produkt GlassFish Server verze 4.
   b) Zadejte příkaz **asadmin** na příkazový řádek.

      Příkaz **asadmin** otevře program shellu na příkazovém řádku, který vám umožňuje vytvořit novou doménu.

   GlassFish Server verze 4 je spuštěn ve vašem systému.
2. Vytvořte a poté spusťte doménu.
   a) Chcete-li vytvořit novou doménu, použijte příkaz **create-domain** s uvedením portu a názvu domény. Na příkazový řádek zadejte následující příkaz:

      ```
      create-domain --adminport port domain_name
      ```

      , kde *port* je číslo portu a *domain_name* je jméno, které chcete, aby doména použila.

      Poznámka: Příkaz **create-domain** má k sobě přiřazeno mnoho volitelných parametrů. Pro tuto úlohu však potřebujete pouze parametr `-- adminport` . Další informace naleznete v dokumentaci produktu GlassFish Server verze 4.

      Je-li zadaný port používán, zobrazí se následující zpráva:

      `Port pro domain_name port se používá`

      Pokud je název domény, který jste uvedli, používán, obdržíte zprávu sdělíte, že zadaný název se již používá, a také seznam všech názvů domén, které jsou momentálně nedostupné.
   b) Když jste vyzváni k zadání jména uživatele a hesla, zadejte pověření, která se mají použít pro přihlášení na aplikační server prostřednictvím webového prohlížeče.

      Pokud je příkaz úspěšně dokončen, zobrazí se na příkazovém řádku zpráva shrnující vytvoření domény včetně zprávy `Command create-domain executed successfully..`

      Úspěšně jste vytvořili doménu.
   c) Spusťte doménu zadáním následujícího příkazu na příkazový řádek:

      ```
      start-domain domain_name
      ```

kde *název_domény* je název domény, kterou jste zadali dříve.

3. Použijte webový prohlížeč pro přístup k aplikačnímu serveru GlassFish .
   a) V adresním řádku webového prohlížeče zadejte tento příkaz:

   ```
   localhost:port
   ```

   kde *port* je port, který jste zadali dříve při vytváření domény.

   Zobrazí se konzola GlassFish .

   b) Když se konzola GlassFish načte a zobrazí se výzva k zadání jména uživatele a hesla, zadejte pověření, která jste zadali v kroku 2b.
4. Odešlete adaptér prostředků na server GlassFish Server 4.
   a) Na panelu nástrojů **Obecné úlohy** vyberte položku nabídky **Aplikace** , abyste zobrazili stránku **Aplikace** .
   b) Klepněte na tlačítko **Implementovat** , abyste otevřeli stránku **Implementovat aplikace nebo moduly** .
   c) Klepněte na tlačítko **Procházet** a poté přejděte do umístění souboru `wmq.jmsra.rar` . Vyberte soubor a klepněte na tlačítko **OK**.
5. Vytvořte fond připojení.
   a) Na panelu nástrojů v části **Prostředky**vyberte položku nabídky **Konektory** .
   b) Poté vyberte položku nabídky **Fondy připojení konektoru** a otevřete stránku **Fondy připojení konektoru** .
   c) Klepnutím na tlačítko **Nový** otevřete stránku **Nový fond připojení konektoru (Krok 1 ze 2)** .
   d) Na stránce **Nový fond připojení konektoru (Krok 1 ze 2)** zadejte název fondu jako jms/ivt/IVTCF-Connection-Fond do pole **Název fondu** .
   e) V poli **Adaptér prostředku** vyberte volbu **wmq.jmsra**.
   f) V poli **Definice připojení** zadejte `javax.jms.ConnectionFactory`.
   g) Vyberte volbu **Další**a poté klepněte na tlačítko **Dokončit**.
6. Vytvořte prostředky konektoru.
   a) Na panelu nástrojů v nabídce **Konektory** vyberte volbu **Prostředek konektoru** , chcete-li otevřít stránku **Prostředky konektoru** .
   b) Vyberte volbu **Nový**, chcete-li otevřít stránku **Nový prostředek konektoru** .
   c) Do pole **Název rozhraní JNDI** zadejte `IVTCF`.
   d) Do pole **Název fondu** zadejte `jms/ivt/IVTCF-Connection-Pool`.
   e) Všechna ostatní pole ponechte prázdná.
   f) Pro každou z následujících dvojic vlastnost/hodnota klepněte na volbu **Přidat vlastnost**a zadejte název vlastnosti a hodnotu tak, jak je uvedeno v následujícím příkladu:
      - název: hostitel; hodnota: localhost
      - název: port; hodnota 1414
      - název: kanál; hodnota: SYSTEM.DEF.SVRCONN
      - název: queueManager; hodnota: QM
      - název: transportType; hodnota: CLIENT

      Poznámka: Ujistěte se, že používáte správné hodnoty pro svá vlastní nastavení konfigurace, která se mohou lišit od hodnot zobrazených v tomto příkladu.
   g) V panelu nástrojů pod položkou **Konektory**vyberte položku nabídky **Prostředky administrativních objektů** a otevřete stránku **Prostředky administrativních objektů** .

h) Na stránce **Prostředky administrativních objektů** klepněte na tlačítko **Nový** a otevřete stránku **Nový prostředek administrativních objektů** .

i) Do pole **Název rozhraní JNDI** zadejte `IVTQueue`.

j) Do pole **Adaptér prostředku** zadejte `wmq.jmsra`.

k) Do pole **Typ prostředku** zadejte `javax.jms.Queue`.

l) Ponechte pole **Název třídy** tak, jak je.

m) Pro každou z následujících dvojic vlastnost/hodnota klepněte na volbu **Přidat vlastnost**a zadejte název vlastnosti a hodnotu tak, jak je uvedeno v následujícím příkladu:

- name: název; hodnota: IVTQueue
- název: baseQueueManagerName; hodnota QM
- název: baseQueueNázev; hodnota: Q1

Poznámka: Ujistěte se, že používáte správné hodnoty pro svá vlastní nastavení konfigurace, která se mohou lišit od hodnot zobrazených v tomto příkladu.

n) Klepněte na tlačítko **OK**.

o) Označte zaškrtávací políčko **Povoleno** , poté klepněte na volbu **Povolit**.

7. Nasaďte soubor EAR `wmq.jmsra.ivt.ear` na server GlassFish Server.

a) Klepněte na volbu **Aplikace** na panelu nástrojů, abyste zobrazili stránku **Aplikace** .

b) Klepnutím na tlačítko **Implementovat** přidejte aplikaci IVT.

c) V poli **Umístění** přejděte na volbu `wmq.jmsra.ivt.ear`a vyberte ji.

d) V poli **Virtual Servers** vyberte volbu **server**a poté klepněte na tlačítko **OK**.

8. Spusťte program IVT.

a) Klepněte na volbu **Aplikace** na panelu nástrojů, abyste zobrazili stránku **Aplikace** .

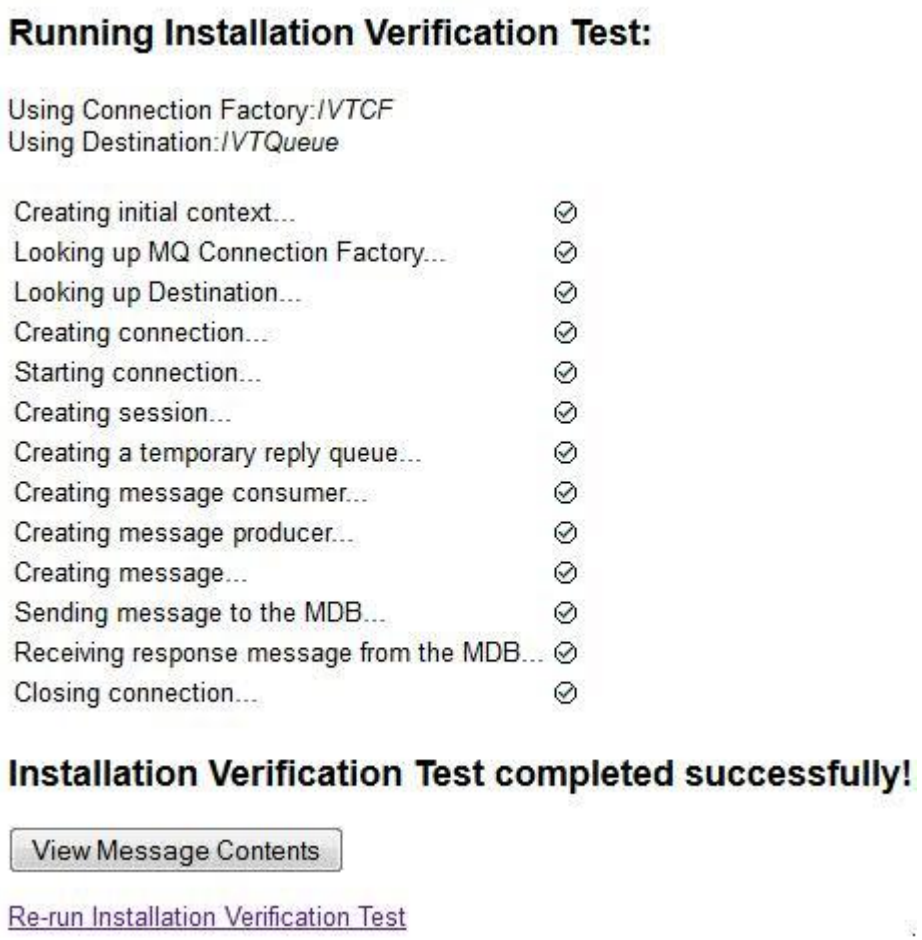
b) Klepněte na `wmq.jmsra.ivt` v tabulce Implementované aplikace.

c) Klepněte na tlačítko **Spustit** v tabulce Moduly a komponenty.

d) Vyberte odkaz http: link.

e) Klepněte na tlačítko **Spustit IVT**.

Vypustili jste program IVT, a pokud jste úspěšní, zobrazí se následující výstup:



Obrázek 55. Úspěšný výstup IVT

V 9.0.2 V 9.0.0.1 Instalace a testování adaptéru prostředků v WildFly

Pokud instalujete adaptér prostředků produktu IBM MQ do adresáře WildFly V10, musíte nejprve provést změny konfiguračního souboru a přidat definici subsystému pro adaptér prostředků produktu IBM MQ . Poté můžete implementovat adaptér prostředků a otestovat jej instalací a spuštěním aplikace testu ověření instalace (IVT).

Informace o této úloze

Důležité: Tyto pokyny jsou určeny pro WildFly V10.

Tato úloha předpokládá, že máte spuštěný aplikační server WildFly a že jste obeznámeni se standardními administračními úlohami pro tuto úlohu. Tato úloha také předpokládá, že máte instalaci produktu IBM MQ a že jste obeznámeni se standardními úlohami administrace.

Postup

1. Vytvořte správce front produktu IBM MQ s názvem ExampleQMa nastavte jej tak, jak je popsáno v tématu “Konfigurace správce front pro příjem klientských připojení na platformách Multiplatforms” na stránce 1045.

   Při nastavování správce front vezměte v úvahu následující body:

   - Modul listener musí být spuštěn na portu 1414.
   - Kanál, který má být použit, se nazývá SYSTEM.DEF.SVRCONN.
   - Fronta používaná aplikací IVT má název TEST.QUEUE.

Modelová fronta SYSTEM.DEFAULT.MODEL.QUEUE je také třeba udělit oprávnění DSP a PUT, takže tato aplikace může vytvořit dočasnou frontu odpovědí.

2. Upravte konfigurační soubor *WildFly_Home*/standalone/configuration/standalone-full.xml a přidejte následující subsystém:

```
<subsystem xmlns="urn:jboss:domain:resource-adapters:4.0">
    <resource-adapters>
        <resource-adapter id="wmq.jmsra">
            <archive>
                wmq.jmsra.rar
            </archive>
            <transaction-support>NoTransaction</transaction-support>
            <connection-definitions>
                <connection-definition class-
name="com.ibm.mq.connector.outbound.ManagedConnectionFactoryImpl"
                                       jndi-name="java:jboss/jms/ivt/IVTCF" enabled="true"
use-java-context="true"
                                       pool-name="IVTCF">
                    <config-property name="channel">SYSTEM.DEF.SVRCONN
                    </config-property>
                    <config-property
                                      name="hostName">localhost
                    </config-property>
                    <config-property name="transportType">
                        CLIENT
                    </config-property>
                    <config-property name="queueManager">
                        ExampleQM
                    </config-property>
                    <config-property name="port">
                        1414
                    </config-property>
                </connection-definition>
                <connection-definition class-
name="com.ibm.mq.connector.outbound.ManagedConnectionFactoryImpl"
                                       jndi-name="java:jboss/jms/ivt/JMS2CF" enabled="true"
use-java-context="true"
                                       pool-name="JMS2CF">
                    <config-property name="channel">
                        SYSTEM.DEF.SVRCONN
                    </config-property>
                    <config-property name="hostName">
                        localhost
                    </config-property>
                    <config-property name="transportType">
                        CLIENT
                    </config-property>
                    <config-property name="queueManager">
                        ExampleQM
                    </config-property>
                    <config-property name="port">
                        1414
                    </config-property>
                </connection-definition>
            </connection-definitions>
        <admin-objects>
            <admin-object class-name="com.ibm.mq.connector.outbound.MQQueueProxy"
                          jndi-name="java:jboss/jms/ivt/IVTQueue" pool-name="IVTQueue">
                <config-property name="baseQueueName">
                    TEST.QUEUE
                </config-property>
                </admin-object>
            </admin-objects>
        </resource-adapter>
    </resource-adapters>
</subsystem>
```

3. Naimplementujte adaptér prostředků na váš server tak, že zkopírujete soubor wmq.jmsra.rar do adresáře *WildFly_Home*/standalone/deployments.
4. Implementujte aplikaci IVT tak, že se urovnávte k souboru wmq.jmsra.ivt.ear do adresáře *WildFly_Home*/standalone/deployments.
5. Spusťte aplikační server tak, že otevřete příkazový řádek, přejdete do adresáře *WildFly_Home*/bin a spustíte příkaz:

```
standalone.bat -c standalone-full.xml
```

6. Spusťte aplikaci IVT.

   Další informace viz “Ověření instalace adaptéru prostředků” na stránce 449. Pro WildFlyje výchozí adresa URL `http://localhost:8080/WMQ_IVT/`.

Společně s IBM MQ a WebSphere Application Server

Through the IBM MQ messaging provider in WebSphere Application Server, Java Message Service (JMS) messaging applications can use your IBM MQ system as an external provider of JMS messaging resources.

Informace o této úloze

Aplikace, které jsou zapsány v produktu Java a jsou spuštěny pod WebSphere Application Server , mohou používat specifikaci Java Messaging Service (JMS) k provádění systému zpráv. Systém zpráv v tomto prostředí může být poskytován správcem front IBM MQ .

Výhodou použití správce front produktu IBM MQ je to, že připojení aplikací produktu JMS se může plně podílet na funkčnosti sítě IBM MQ , což umožňuje aplikacím vyměňovat si zprávy se správci front spuštěnými na velkém počtu platforem.

Aplikace mohou pro objekt továrny připojení fronty použít buď *přenos klienta* , nebo *přenos vazeb* . Pro přenos vazeb musí správce front existovat lokálně na aplikaci, která vyžaduje připojení.

Zprávy produktu JMS ve frontách produktu IBM MQ standardně používají záhlaví MQRFH2 k ukládání některých informací záhlaví zprávy produktu JMS . Mnoho starších aplikací produktu IBM MQ nemůže zpracovávat zprávy s těmito záhlavími a vyžadovat jejich vlastní záhlaví charakteristik, například MQCIH pro most CICS Bridge nebo MQWIH pro aplikace IBM MQ Workflow. Další informace o těchto speciálních aspektech naleznete v tématu Mapování zpráv produktu JMS na zprávy produktu IBM MQ.

Související informace

Konfigurace prostředků produktu JMS v produktu WebSphere Application Server

Konfigurace aplikačního serveru pro použití nejnovější úrovně údržby adaptéru prostředků

Použití WebSphere Application Server s IBM MQ

IBM MQ a IBM MQ for z/OS lze použít spolu s výchozím poskytovatelem systému zpráv, který je součástí produktu WebSphere Application Server, nebo jako alternativa k výchozímu poskytovateli systému zpráv.

Poskytovatel systému zpráv produktu IBM MQ je nainstalován jako součást produktu WebSphere Application Server. Jedná se o verzi adaptéru prostředků produktu IBM MQ a funkce rozšířených transakčních klientů produktu IBM MQ , která umožňuje správci front podílet se na transakcích XA spravovaných aplikačním serverem. Pomocí adaptéru prostředků lze objekty typu message-driven bean konfigurovat tak, aby používaly buď specifikace aktivace, nebo porty modulu listener.

Aby mohl být aplikační server podporován, je třeba produkt Ověřovací testovací program pro instalaci adaptéru prostředků produktu IBM MQ implementovat na aplikační server a úspěšně jej spustit. Po úspěšném spuštění testovacího programu pro ověření adaptéru prostředků produktu IBM MQ se adaptér prostředků produktu IBM MQ může připojit k libovolnému podporovanému správci front IBM MQ .

Připojení produktu JMS z WebSphere Application Server na IBM MQ

Před zvažováním úrovní IBM MQ , které lze použít s produktem WebSphere Application Server, je důležité pochopit, jak se aplikace produktu Java Message Service (JMS) spuštěné v rámci aplikačního serveru mohou připojovat ke správcům front IBM MQ .

Aplikace produktu JMS , které potřebují přistupovat k prostředkům správce front produktu IBM MQ , mohou tak učinit pomocí jednoho z následujících typů transportu:

Vazby

Tento přenos lze použít, je-li na stejném počítači a v obrazu operačního systému nainstalován aplikační server a správce front. Při použití režimu BINDINGS se veškerá komunikace mezi dvěma produkty provádí pomocí komunikace IPC (Inter-Process Communication).

Poskytovatel systému zpráv produktu IBM MQ nezahrnuje nativní knihovny vyžadované pro připojení ke správci front produktu IBM MQ v režimu BINDINGS. Chcete-li použít připojení režimu BINDINGS, produkt IBM MQ musí být nainstalován na stejném počítači jako aplikační server a cesta k nativní knihovně adaptéru prostředku musí být konfigurována tak, aby ukazovala na adresář IBM MQ , kde jsou tyto knihovny umístěny. Další informace najdete v dokumentaci produktu WebSphere Application Server :

- Informace o produktu WebSphere Application Server traditionalnaleznete v části Konfigurace poskytovatele systému zpráv produktu IBM MQ s použitím nativních knihoven.
- Informace o produktu WebSphere Application Server Libertynaleznete v tématu Implementace aplikací JMS do Liberty pro použití poskytovatele systému zpráv produktu IBM MQ.

z/OS Chcete-li v produktu z/OSpřipojit továrnu připojení produktu WebSphere Application Server k správci front produktu IBM MQ v režimu vazeb, je třeba určit správné knihovny produktu IBM MQ ve zřetězení STEPLIB produktu WebSphere Application Server . Další informace naleznete v tématu Knihovny produktuIBM MQ a WebSphere Application Server pro z/OS STEPLIB v dokumentaci produktu WebSphere Application Server .

CLIENT
Přenos klienta používá TCP/IP ke komunikaci mezi WebSphere Application Server a IBM MQ. Je-li aplikační server a správce front umístěny na různých počítačích, lze použít také režim CLIENT, pokud jsou tyto dva produkty nainstalovány na stejném počítači a v obrazu operačního systému.

Aplikace produktu JMS mohou také určovat typ transportu BINDINGS_THEN_CLIENT. Je-li použit tento typ transportu, aplikace se nejprve pokusí o připojení ke správci front pomocí režimu BINDINGS-pokud to není možné, pokusí se o přenos CLIENT.

Jak zjistit, která verze adaptéru prostředků IBM MQ je instalována uvnitř WebSphere Application Server

Informace o tom, která verze adaptéru prostředků IBM MQ je instalována v produktu WebSphere Application Server, naleznete v technické poznámce Jaká verze adaptéru prostředků WebSphere MQ (RA) je dodávána se serverem WebSphere Application Server?.

Chcete-li určit úroveň adaptéru prostředků, který produkt WebSphere Application Server aktuálně používá, můžete použít následující příkazy jazyka Jython a JACL:

Jython

```
wmqInfoMBeansUnsplit = AdminControl.queryNames("WebSphere:type=WMQInfo,*")
wmqInfoMBeansSplit = AdminUtilities.convertToList(wmqInfoMBeansUnsplit)
for wmqInfoMBean in wmqInfoMBeansSplit: print wmqInfoMBean; print
AdminControl.invoke(wmqInfoMBean, 'getInfo', '')
```

Poznámka: Musíte klepnout na tlačítko **Návrat** dvakrát po zadání tohoto příkazu, abyste jej mohli spustit.

JACL.

```
set wmqInfoMBeans [$AdminControl queryNames WebSphere:type=WMQInfo,*]
foreach wmqInfoMBean $wmqInfoMBeans {
puts $wmqInfoMBean;
puts [$AdminControl invoke $wmqInfoMBean getInfo [] []]
}
```

Aktualizace adaptéru prostředků

Aktualizace adaptéru prostředků produktu IBM MQ , který je instalován spolu s aplikačním serverem, jsou obsaženy v opravných sadách produktu WebSphere Application Server . Aktualizace adaptéru prostředků produktu IBM MQ pomocí volby **Aktualizovat adaptér prostředků ...** prostředek v administrativní konzole

WebSphere Application Server se nedoporučuje, protože to znamená, že aktualizace poskytované v sadě WebSphere Application Server Fix Pack nebudou mít žádný vliv.

proměnná MQ_INSTALL_ROOT

WebSphere Application Server verze starší než 7.0 mohly být nakonfigurovány tak, aby používaly IBM WebSphere MQ classes for JMS umístěný v externí instalaci produktu IBM WebSphere MQ pro připojení ke správci front nastavením proměnné WebSphere MQ_INSTALL_ROOT.

Z produktu WebSphere Application Server 7.0se hodnota MQ_INSTALL_ROOT používá pouze k vyhledání nativních knihoven a je potlačena jakoukoli cestou k nativní knihovně, která je konfigurována na adaptéru prostředků.

Připojení z WebSphere Application Server do IBM MQ

Upozornění:

1. Jakákoli podporovaná verze produktu WebSphere Application Server může použít adaptér prostředků produktu IBM MQ , který je součástí balíku, pro připojení k jakékoli podporované verzi produktu IBM MQ.
2. Je-li použit režim vazeb, určité knihovny v produktu WebSphere Application Server se musí shodovat s verzí správce front, ke kterému se připojuje:
   - Produkt WebSphere Application Server musí být nakonfigurován k načtení nativních knihoven poskytnutých s produktem IBM MQ 9.0. Další informace viz “Konfigurace knihoven JNI (Java Native Interface)” na stránce 79.
   - z/OS V systému z/OSje třeba určit správné knihovny produktu IBM MQ ve zřetězení STEPLIB produktu WebSphere Application Server .

     Podrobnosti o knihovnách produktu IBM MQ , které potřebujete, najdete v tématu Knihovny produktuIBM MQ a knihovna WebSphere Application Server for z/OS STEPLIB .

     Máte-li knihovny pro jednu verzi produktu IBM MQ v datové sadě LINKLIST (LINKLST), můžete se připojit k jiné verzi produktu IBM MQ přepisováním knihoven STEPLIB.
3. Verze adaptéru prostředků produktu IBM MQ je nezávislá na nativních (sdílených) verzích knihovny, které jsou poskytovány instalací správce front.

   Například WebSphere Application Server 8.5s adaptérem prostředků IBM WebSphere MQ 7.1 může stále spravovat vazby připojení ke správci front IBM MQ 9.0 pomocí nativních knihoven IBM MQ 9.0 .

Další informace viz “Příkaz podpory adaptéru prostředků produktu IBM MQ” na stránce 402.

Následující tabulka ukazuje, které typy transportu lze použít pro připojení k produktu IBM MQ ze všech verzí produktu WebSphere Application Server.

Verze IBM MQ nebo IBM WebSphere MQ	Přenos BINDINGS	Přenos CLIENT
IBM MQ 9.0	Podporováno. • Produkt IBM MQ 9.0 musí být nainstalován na stejném počítači jako aplikační server. • Produkt WebSphere Application Server musí být nakonfigurován k načtení nativních knihoven	Podporováno

Verze IBM MQ nebo IBM WebSphere MQ	Přenos BINDINGS	Přenos CLIENT
	poskytnutých s produktem IBM MQ 9.0. • z/OS Chcete-li v produktu z/OSpřipojit továrnu připojení produktu WebSphere Application Server ke správci front produktu IBM MQ v režimu vazeb, musí být ve zřetězení STEPLIB příkazu WebSphere Application Server uvedeny správné knihovny produktu IBM MQ .	
IBM MQ 8.0	Podporováno. • Produkt IBM MQ 8.0 musí být nainstalován na stejném počítači jako aplikační server. • Produkt WebSphere Application Server musí být nakonfigurován k načtení nativních knihoven poskytnutých s produktem IBM MQ 8.0. • z/OS Chcete-li v produktu z/OSpřipojit továrnu připojení produktu WebSphere Application Server ke správci front produktu IBM MQ v režimu vazeb, musí být ve zřetězení STEPLIB příkazu WebSphere Application Server uvedeny správné knihovny produktu IBM MQ .	Podporováno
IBM WebSphere MQ 7.5	Podporováno. • Produkt IBM WebSphere MQ 7.5 musí být nainstalován na stejném počítači jako aplikační server. • Produkt WebSphere Application Server musí být nakonfigurován k načtení nativních knihoven poskytnutých s produktem IBM WebSphere MQ 7.5. • z/OS Chcete-li v produktu z/OSpřipojit továrnu připojení produktu WebSphere Application Server k správce	Podporováno

Verze IBM MQ nebo IBM WebSphere MQ	Přenos BINDINGS	Přenos CLIENT
	front produktu IBM WebSphere MQ v režimu vazeb, je třeba v zřetězení STEPLIB WebSphere Application Server zadat správné knihovny produktu IBM WebSphere MQ .	
IBM WebSphere MQ 7.1	Podporováno. • Produkt IBM WebSphere MQ 7.1 musí být nainstalován na stejném počítači jako aplikační server. • Produkt WebSphere Application Server musí být nakonfigurován k načtení nativních knihoven poskytnutých s produktem IBM WebSphere MQ 7.1. • z/OS Chcete-li v produktu z/OSpřipojit továrnu připojení produktu WebSphere Application Server k správce front produktu IBM WebSphere MQ v režimu vazeb, je třeba v zřetězení STEPLIB WebSphere Application Server zadat správné knihovny produktu IBM WebSphere MQ .	Podporováno

V následující tabulce jsou uvedeny verze produktu WebSphere Application Server , pro které je podporováno spuštění adaptéru prostředků produktu IBM MQ .

Verze adaptéru prostředků produktu IBM MQ	Kterou verzi produktu WebSphere Application Server lze v této verzi adaptéru prostředků spustit?
IBM MQ 9.0	Adaptér prostředku může být spuštěn v: • Any Java EE 7 compliant version of WebSphere Application Server Liberty. • WebSphere Application Server traditional 9.0
IBM MQ 8.0	The resource adapter can run in any Java EE 7 compliant version of WebSphere Application Server Liberty Adaptér prostředků IBM MQ 8.0 není podporován pro spuštění v produktu WebSphere Application Server traditional. Adaptér prostředků již instalovaný v produktu WebSphere Application Server traditional by měl být použit pro připojení ke správcům front produktu IBM MQ 8.0 .

Verze adaptéru prostředků produktu IBM MQ	Kterou verzi produktu WebSphere Application Server lze v této verzi adaptéru prostředků spustit?
IBM WebSphere MQ 7.5	Adaptér prostředků lze použít v aplikačních serverech J2EE 1.4 nebo novějších. V těchto prostředích by měla být použita verze adaptéru prostředků produktu IBM WebSphere MQ , která je obsažena v produktu WebSphere Application Server 8.0 a 7.0 . IBM WebSphere MQ 7.5.0 Fix Pack 2 a oprava APAR IC92914 přidávají podporu pro implementaci adaptéru prostředků do produktu WebSphere Application Server Liberty.
IBM WebSphere MQ 7.1	Adaptér prostředků lze použít v aplikačních serverech J2EE 1.4 nebo novějších. V těchto prostředích by měla být použita verze adaptéru prostředků produktu IBM WebSphere MQ , která je obsažena v produktu WebSphere Application Server 8.0 a 7.0 .

Související pojmy

“Příkaz podpory adaptéru prostředků produktu IBM MQ” na stránce 402
Adaptér prostředků, který je dodáván s produktem IBM MQ 8.0 nebo novějším, implementuje specifikaci JMS 2.0 . Lze jej implementovat pouze na aplikační server, který je kompatibilní se systémem Java Platform, Enterprise Edition 7 (Java EE 7), a proto podporuje JMS 2.0.

Související informace

Systémové požadavky pro IBM MQ

Určení počtu připojení TCP/IP vytvořených z produktu WebSphere Application Server do produktu IBM MQ

Produkt IBM WebSphere MQ 7.0 představil novou funkci nazvanou "sdílení konverzací". Pomocí této funkce může více konverzací sdílet instance kanálu MQI, což je také známé jako připojení TCP/IP.

Informace o této úloze

Aplikace běžící uvnitř WebSphere Application Server 7 a 8, které používají poskytovatele systému zpráv IBM MQ v normálním režimu, tuto funkci automaticky použijí. To znamená, že více aplikací spuštěných v rámci jedné instance aplikačního serveru, které se připojují ke stejnému správci front produktu IBM MQ , jsou schopny sdílet stejnou instanci kanálu.

Počet konverzací, které lze sdílet v rámci jedné instance kanálu, je určen vlastností kanálu IBM MQ **SHARECNV**. Výchozí hodnota této vlastnosti pro kanály připojení serveru je 10.

Při pohledu na počet konverzací, které byly vytvořeny WebSphere Application Server 7 a 8, je možné určit počet vytvořených instancí kanálu.

Další informace o režimu poskytovatele systému zpráv produktu IBM MQ naleznete v normálním režimu PROVIDERVERSION.

Související pojmy

Použití sdílení konverzací
V prostředí, ve kterém je povoleno sdílení konverzací, mohou konverzace sdílet instanci kanálu MQI.

“Sdílení připojení TCP/IP v produktu IBM MQ classes for JMS” na stránce 284
Je možné vytvořit více instancí kanálu MQI, aby bylo možné sdílet jedno připojení TCP/IP.

Továrny připojení JMS

Aplikace spuštěné v rámci produktu WebSphere Application Server, které používají továrnu připojení poskytovatele systému zpráv produktu IBM MQ k vytváření připojení a relací, mají aktivní konverzace pro každé připojení JMS vytvořené z továrny připojení a pro každou relaci JMS vytvořenou z připojení produktu JMS .

Jedna konverzace pro každé připojení produktu JMS , které bylo vytvořeno z továrny připojení

Každá továrna připojení produktu JMS má přidružený fond připojení, rozdělený na dvě sekce, volný fond a aktivní fond. Oba fondy jsou na počátku prázdné.

Když aplikace vytvoří připojení produktu JMS z továrny připojení, příkaz WebSphere Application Server zkontroluje, zda je ve volném fondu připojení JMS . Pokud existuje, přesune se do aktivního fondu a dostane se do aplikace. Jinak se vytvoří nové připojení JMS , vloží se do aktivního fondu a vrátí se do aplikace. Maximální počet připojení, které lze vytvořit z továrny připojení, je určen vlastností fondu připojení továrny připojení **Maximum connections**. Výchozí hodnota této vlastnosti je 10.

Po dokončení aplikace s připojením produktu JMS a zavřením se připojení přesune z aktivního fondu do volného fondu, kde je k dispozici pro opětovné použití. Vlastnost fondu připojení **Unused timeout** definuje, jak dlouho může JMS připojení zůstat ve volném fondu, než se odpojí. Výchozí hodnota této vlastnosti je 1800 sekund (30 minut).

Při prvním vytvoření připojení produktu JMS se spustí konverzace mezi produkty WebSphere Application Server a IBM MQ . Konverzace zůstane aktivní, dokud se připojení nezavře, když je překročena hodnota vlastnosti **Unused timeout** pro volný fond.

Jedna konverzace pro každou relaci JMS , která byla vytvořena z připojení JMS

Každé připojení produktu JMS , které je vytvořeno z továrny připojení poskytovatele systému zpráv produktu IBM MQ , má přidružený fond relací JMS . Fondy relací pracují stejným způsobem jako fondy připojení. Maximální počet relací JMS , které lze vytvořit z jednoho připojení JMS , je určen vlastností fondu relací továrny připojení **Maximum connections**. Výchozí hodnota této vlastnosti je 10.

Konverzace se spustí, když je nejprve vytvořena relace JMS , konverzace zůstává aktivní, dokud se relace JMS neuzavře, protože zůstala ve volném fondu déle, než hodnota vlastnosti **Unused timeout** pro fond relací.

Výpočet hodnoty vlastnosti SHARECNV

Maximální počet konverzací z jedné továrny připojení na produkt IBM MQ můžete vypočítat pomocí následujícího vzorce:

```
Maximum number of conversations =
    connection Pool Maximum Connections +
   (connection Pool Maximum Connections * Session Pool Maximum Connections)
```

Počet instancí kanálu, které budou vytvořeny tak, aby umožňoval tento počet konverzací, může být proveden pomocí následujícího výpočtu:

```
Maximum number of channel instances =
    Maximum number of conversations / SHARECNV for the channel being used
```

Jakýkoli zůstatek z tohoto výpočtu lze zaokrouhlit nahoru.

Pro jednoduchou továrnu připojení, která používá výchozí hodnotu pro fond připojení **Maximum connections** a vlastnosti fondu relací **Maximum connections** , je maximální počet konverzací, které mohou existovat mezi WebSphere Application Server a IBM MQ pro tuto továrnu připojení, následující:

```
Maximum number of conversations =
```

```
   connection Pool Maximum Connections +
  (connection Pool Maximum Connections * Session Pool Maximum Connections)
```

Příklad:

```
        = 10 + (10 * 10)
        = 10 + 100
        = 110
```

Pokud se tato továrna připojení připojuje k produktu IBM MQ pomocí kanálu s nastavenou vlastností **SHARECNV** na hodnotu 10, bude maximální počet instancí kanálu, které budou vytvořeny pro tuto továrnu připojení, následující:

```
Maximum number of channel instances = Maximum number of conversations / SHARECNV for the
channel being used
```

Příklad:

```
        = 110 / 10
        = 11 (rounded up to nearest connection)
```

Specifikace aktivace

Aplikace typu message-driven bean, které jsou konfigurovány k použití specifikace aktivace, mají konverzace aktivní pro specifikaci aktivace pro monitorování cíle JMS a pro každou relaci serveru používanou ke spuštění instance objektu bean řízeného zprávami ke zpracování zpráv.

Následující konverzace jsou aktivní pro aplikace objektu typu message driven bean, které jsou konfigurovány pro použití specifikace aktivace:

- Jedna konverzace pro specifikaci aktivace pro monitorování cíle JMS pro vhodné zprávy. Tato konverzace se spustí ihned po spuštění specifikace aktivace a zůstane aktivní, dokud se specifikace aktivace nezastaví.
- Jedna konverzace pro každou relaci serveru, která se používá ke spuštění instance objektu typu message driven bean ke zpracování zpráv.

Rozšířená vlastnost specifikace aktivace **Maximum server sessions** určuje maximální počet relací serveru, které mohou být aktivní v libovolné chvíli pro danou specifikaci aktivace. Tato vlastnost má výchozí hodnotu 10. Relace serveru se vytvářejí podle potřeby a jsou zavřeny, pokud byly nečinné po dobu určenou rozšířenou vlastností specifikace aktivace **Server session pool timeout**. Výchozí hodnota této vlastnosti je 300000 milisekund (5 minut).

Konverzace se spouští při vytvoření relace serveru a jsou zastaveny buď při zastavení specifikace aktivace, nebo při vypršení časového limitu relace serveru.

To znamená, že maximální počet konverzací z jedné specifikace aktivace do IBM MQ může být vypočítán pomocí následujícího vzorce:

```
Maximum number of conversations = Maximum server sessions + 1
```

Počet instancí kanálu, které jsou vytvořeny, aby umožňoval tento počet konverzací, lze nalézt pomocí následujícího výpočtu:

```
Maximum number of channel instances =
    Maximum number of conversations / SHARECNV for the channel being used
```

Jakýkoli zůstatek z tohoto výpočtu lze zaokrouhlit nahoru.

Pro jednoduchou aktivační specifikaci, která používá výchozí hodnotu vlastnosti **Maximum server sessions** , se maximální počet konverzací, které mohou existovat mezi WebSphere Application Server a IBM MQ pro tuto specifikaci aktivace, vypočítá jako:

```
Maximum number of conversations = Maximum server sessions + 1
```

Příklad:

```
    = 10 + 1
    = 11
```

Pokud se tato specifikace aktivace připojuje k produktu IBM MQ pomocí kanálu, který má nastavovanou vlastnost **SHARECNV** na hodnotu 10, bude počet vytvořených instancí kanálu vypočten takto:

```
Maximum number of channel instances =
    Maximum number of conversations / SHARECNV for the channel being used
```

Příklad:

```
    = 11 / 10
    = 2 (rounded up to nearest connection)
```

Porty modulu listener spuštěné v režimu ASF (Application Server Facilities)

Porty modulu listener spuštěné v režimu ASF využívaného aplikacemi typu message-driven bean vytvářejí konverzace pro každou relaci serveru. Jeden monitoruje místo určení pro vhodné zprávy a jiná spustí instanci objektu typu message-driven bean ke zpracování zpráv. Počet konverzací pro každý port modulu listener lze vypočítat z maximálního počtu relací.

Porty modulu listener standardně běží v režimu ASF jako součást specifikace 1.1 , která definuje mechanizmus, který by aplikační servery měly používat ke zjišťování zpráv a jejich doručení do objektů typu message-driven bean ke zpracování. Aplikace typu message-driven bean, které jsou nastaveny tak, aby používaly porty modulu listener v tomto výchozím režimu operací vytvoření konverzací:

Jedna konverzace pro port modulu listener k monitorování místa určení pro vhodné zprávy
: Porty modulu listener jsou konfigurovány tak, aby používaly továrnu připojení produktu JMS . Při spuštění portu modulu listener dojde k požadavku na připojení produktu JMS z fondu volných prostředků továrny připojení. Když je port modulu listener zastaven, je připojení vráceno do fondu volných prostředků. Další informace o způsobu použití fondu připojení a o tom, jak tato hodnota ovlivňuje počet konverzací v produktu IBM MQ, najdete v tématu “Továrny připojení JMS” na stránce 463.

Jedna konverzace pro každou relaci serveru, která se používá ke spuštění instance objektu typu message driven bean ke zpracování zpráv
: Vlastnost portu modulu listener **Maximum sessions** určuje maximální počet relací serveru, které mohou být v jednom okamžiku pro daný port modulu listener aktivní. Tato vlastnost má výchozí hodnotu 10. Relace serveru jsou vytvářeny podle potřeby a využívají relace JMS převzaté ze společné oblasti relace přidružené k připojení produktu JMS , které používá port modulu listener.

Byla-li relace serveru nečinná po dobu určenou přizpůsobenou vlastností služby listener pro zprávy **SERVER.SESSION.POOL.UNUSED.TIMEOUT**, relace se zavře a relace JMS se vrátí do fondu volných fondů relací. Relace JMS zůstane ve fondu volných prostředků fondu relací, dokud není potřeba, nebo je zavřena, protože byla nečinná ve volném fondu déle, než je hodnota vlastnosti **Unused timeout** fondu relací.

Další informace o způsobu použití fondu relací a o tom, jak jsou spravovány konverzace mezi WebSphere Application Server a IBM MQ , viz “Továrny připojení JMS” na stránce 463.

Další informace o přizpůsobené vlastnosti služby listener pro zprávy **SERVER.SESSION.POOL.UNUSED.TIMEOUT**, viz Monitorování fondů relací serveru pro porty modulu listener v dokumentaci produktu WebSphere Application Server .

Výpočet maximálního počtu konverzací z jednoho portu modulu listener na produkt IBM MQ

Maximální počet konverzací z jednoho portu modulu listener na produkt IBM MQ můžete vypočítat pomocí následujícího vzorce:

```
Maximum number of conversations = Maximum sessions + 1
```

Počet instancí kanálu, které budou vytvořeny tak, aby umožňoval tento počet konverzací, může být proveden pomocí následujícího výpočtu:

```
     Maximum number of channel instances =
        Maximum number of conversations / SHARECNV for the channel being used
```

Jakýkoli zůstatek z tohoto výpočtu lze zaokrouhlit nahoru.

Pro jednoduchý port modulu listener, který používá výchozí hodnotu vlastnosti **Maximum sessions** , se maximální počet konverzací, které mohou existovat mezi WebSphere Application Server a IBM MQ pro tento port modulu listener, vypočítá jako:

```
Maximum number of conversations = Maximum sessions + 1
```

Příklad:

```
    = 10 + 1
    = 11
```

Pokud se tento port modulu listener připojuje k produktu IBM MQ pomocí kanálu, který má nastavovanou vlastnost **SHARECNV** na hodnotu 10, bude počet instancí kanálu, které budou vytvořeny, vypočten takto:

```
Maximum number of channel instances =
    Maximum number of conversations / SHARECNV for the channel being used
```

Příklad:

```
    = 11 / 10
    = 2 (rounded up to nearest connection)
```

Porty modulu listener spuštěné v režimu bez použití ASF (Application Server Facilities)

Porty modulu listener spuštěné v režimu non-ASF mohou být konfigurovány tak, aby monitorovaly cíl fronty a cíl tématu pomocí relací serveru. Relace serveru mohou mít více konverzací, přičemž maximální počet může být vypočítán v každém jednotlivém případě.

Porty modulu listener lze konfigurovat tak, aby se spouštěly v režimu non-ASF, který změní způsob, jakým porty modulu listener monitorují JMS místa určení. Aplikace typu message-driven bean, které používají porty modulu listener v režimu non-ASF operace, vytvářejí konverzaci pro každou relaci serveru používanou ke spuštění instance objektu bean řízeného zprávami ke zpracování zpráv. Vlastnost portu modulu listener **maximum sessions** určuje maximální počet relací serveru, které mohou být v daném portu modulu listener aktivní v jednom okamžiku. Výchozí hodnota této vlastnosti je 10.

Při spuštění v jiném režimu než ASF bude cílem portu modulu listener pro cíl fronty automaticky vytvořit počet relací serveru určených vlastností portu modulu listener **Maximální počet relací**. Všechny tyto relace serveru využívají relace produktu JMS , které jsou převzaty z fondu relací přidruženého k připojení JMS , které port modulu listener používá, a nepřetržitě sleduje cíl JMS pro vhodné zprávy.

Je-li port modulu listener nakonfigurován pro monitorování místa určení tématu, je hodnota **Maximální počet relací** ignorována a je použita jedna relace serveru.

Relace serveru používané portem modulu listener spouštěným v režimu non-ASF zůstávají aktivní, dokud není zastaven port modulu listener. V tomto bodě jsou použity relace JMS , které byly použity, do fondu volných prostředků fondu relací pro JMS Connection, které port modulu listener používal.

Další informace o způsobu použití fondu relací a o tom, jak jsou spravovány konverzace mezi WebSphere Application Server a IBM MQ , viz “Továrny připojení JMS” na stránce 463.

Další informace o režimu ASF a non-ASF s produktem WebSphere Application Servera o tom, jak nakonfigurovat porty modulu listener pro použití jiného režimu než ASF, naleznete v tématu Zpracování zpráv v režimu ASF a v režimu non-ASF.

Výpočet maximálního počtu konverzací při monitorování místa určení fronty

Maximální počet konverzací z jednoho portu modulu listener, spuštěného v režimu non-ASF a monitorování cíle fronty do produktu IBM MQ , lze vypočítat pomocí následujícího vzorce:

```
Maximum number of conversations = Maximum sessions
```

Počet instancí kanálu, které budou vytvořeny pro umožnění tohoto počtu konverzací, které se mají provést, lze nalézt pomocí následujícího výpočtu:

```
Maximum number of channel instances =
   Maximum number of conversations / SHARECNV for the channel being used
```

Jakýkoli zůstatek z tohoto výpočtu lze zaokrouhlit nahoru.

Pro jednoduchý port modulu listener spuštěný v jiném režimu než ASF, který používá výchozí hodnotu pro vlastnost **Maximální počet relací** a monitorování cíle fronty, je maximální počet konverzací, které mohou existovat mezi WebSphere Application Server a IBM MQ pro tento port modulu listener:

```
Maximum number of conversations = Maximum sessions
```

Příklad:

```
= 10
```

Pokud se tento port modulu listener připojuje k produktu IBM MQ s použitím kanálu, který má nastavovanou vlastnost **SHARECNV** na hodnotu 10, bude počet vytvořených instancí kanálu vypočítán takto:

```
Maximum number of channel instances =
   Maximum number of conversations / SHARECNV for the channel being used
```

Příklad:

```
= 10 / 10
= 1
```

Výpočet maximálního počtu konverzací při monitorování místa určení tématu

Pro port modulu listener spuštěný v režimu non-ASF a konfigurovaný pro monitorování místa určení tématu je počet konverzací z portu modulu listener na IBM MQ následující:

```
Maximum number of conversations = 1
```

Počet instancí kanálu, které budou vytvořeny pro umožnění tohoto počtu konverzací, které se mají provést, lze nalézt pomocí následujícího výpočtu:

```
Maximum number of channel instances =
   Maximum number of conversations / SHARECNV for the channel being used
```

Jakýkoli zůstatek z tohoto výpočtu lze zaokrouhlit nahoru.

Pro jednoduchý port modulu listener spuštěný v jiném režimu než ASF, který používá výchozí hodnotu pro vlastnost **Maximum relací** a monitorování cíle tématu, je maximální počet konverzací, které mohou existovat mezi WebSphere Application Server a IBM MQ pro tento port modulu listener:

```
Maximum number of conversations = Maximum sessions
```

Příklad:

```
= 10
```

Pokud se tento port modulu listener připojuje k produktu IBM MQ s použitím kanálu, který má nastavovanou vlastnost **SHARECNV** na hodnotu 10, bude počet vytvořených instancí kanálu vypočítán takto:

```
Maximum number of channel instances =
   Maximum number of conversations / SHARECNV for the channel being used
```

Příklad:

```
= 10 / 10
= 1
```

Použít aliasy ověřování pro zabezpečení připojení produktu WebSphere Application Server k produktu IBM MQ

Aliasy ověřování jsou mapovány na kombinaci jména uživatele a hesla, která lze použít k zabezpečení připojení produktu WebSphere Application Server k produktu IBM MQ. Továrnu připojení lze konfigurovat s použitím aliasu ověřování.

Použití aliasů ověřování v podnikových aplikacích

Pokud podniková aplikace spuštěná v produktu WebSphere Application Server se pokusí vytvořit připojení produktu JMS k produktu IBM MQ, vyhledá aplikace definici továrny připojení poskytovatele systému zpráv produktu IBM MQ z úložiště Java Naming Directory Interface (JNDI) aplikačního serveru.

Je-li definice továrny připojení poskytovatele systému zpráv produktu IBM MQ umístěna v úložišti produktu JNDI na aplikačním serveru, je volána jedna z následujících metod:

- `ConnectionFactory.createConnection()`
- `ConnectionFactory.createConnection(String username, String password)`

Pokud byla továrna připojení konfigurována s použitím aliasu ověřování J2C , lze při použití továrny připojení k vytvoření připojení proudí jméno uživatele a heslo v aliasu ověřování do produktu IBM MQ .

Továrny připojení a aliasy ověřování

Továrny připojení poskytovatele systému zpráv produktu IBM MQ obsahují informace o tom, jak se připojit ke správcům front produktu IBM MQ . Podnikové aplikace spuštěné v rámci produktu WebSphere Application Server mohou používat továrny připojení k vytváření připojení produktu JMS k produktu IBM MQ.

Produkt WebSphere Application Server ukládá definice továren připojení v úložišti, ke kterému lze přistupovat prostřednictvím produktu JNDI. Při vytvoření továrny připojení je k této továrně připojení

poskytnut název produktu JNDI , který ji jednoznačně identifikuje v oboru aplikačního serveru (obor buňky, Uzel nebo Server), ve kterém byla definována.

Příklad: Továrna připojení poskytovatele systému zpráv produktu IBM MQ definovaná v oboru buňky produktu WebSphere Application Server obsahuje informace o tom, jak se připojit ke správci front (`myQM`) pomocí přenosu BINDINGS. Této továrně připojení je přidělen název JNDI `jms/myCF` , který ji jednoznačně identifikuje.

Továrny připojení lze také nakonfigurovat tak, aby používaly alias ověřování. Aliasy ověřování se mapují na kombinaci jména uživatele a hesla. V závislosti na způsobu použití továrny připojení může být jméno uživatele a heslo v aliasu pro ověření nebo nemusí být při vytvoření připojení k produktu JMS odtéráno do produktu IBM MQ .

Důležité: Před IBM MQ 8.0provedl výchozí produkt IBM MQ Object Authority Manager (OAM) kontrolu autorizace, pouze aby bylo zajištěno, že jméno uživatele předané produktu IBM MQpři připojení má oprávnění pro přístup ke správci front.

Nebyly provedeny žádné kontroly k ověření platnosti hesla, které bylo uvedeno. Chcete-li provést kontrolu ověření a ověřit, že se shoduje identifikátor uživatele a heslo, musíte napsat uživatelskou proceduru pro zabezpečení kanálu produktu IBM MQ . Podrobnosti o tom, jak to lze provést, najdete v tématu Uživatelské programy zabezpečení kanálu.

V produktu IBM MQ 8.0správce front kontroluje heslo spolu se jménem uživatele.

Použití továrny připojení

Následující témata obsahují informace o použití továrny připojení s použitím přímého a nepřímého vyhledávání:

- “Použití továrny připojení pomocí přímého vyhledávání” na stránce 472
- “Použití továrny připojení prostřednictvím nepřímého vyhledávání” na stránce 473

Použití přenosu CLIENT

Továrny připojení, které jsou konfigurovány k použití přenosu CLIENT, musí určovat, který kanál připojení serveru IBM MQ (SVRCONN) se bude používat pro připojení ke správci front.

Zůstane-li vlastnost MCAUSER (identifikátor uživatele) kanálu IBM MQ (MCAUSER) prázdná pro kanál, který byla konfigurována pro použití, lze továrnu připojení použít buď s přímým pohledem nahoru, nebo s nepřímou kontrolou.

Je-li vlastnost MCAUSER nastavena na identifikátor uživatele, je tento identifikátor uživatele předáván do produktu IBM MQ v případě, že se továrna připojení používá k vytvoření připojení k produktu IBM MQbez ohledu na to, zda podniková aplikace používá přímé nebo nepřímé vyhledávání.

souhrnné tabulky

Následující tabulky shrnují, které identifikátory uživatelů jsou při přenosu BINDINGS a přenos CLIENT použity k produktu IBM MQ , a to:

Tabulka 70. Režim BINDINGS

Konfigurace	**Volání aplikace `ConnectionFactory.createConnection()`**	**Volání aplikace `ConnectionFactory.createConnection(String username, String password)`**
Deskriptor implementace aplikace neobsahuje odkaz na prostředek pro továrnu připojení.	Identifikátor uživatele pro proces aplikačního serveru se tečí do IBM MQ.	Identifikátor uživatele a heslo, které byly předány do metody `ConnectionFactory.createConnection(String username, String password)` , jsou předávány do IBM MQ.
Deskriptor implementace aplikace obsahuje odkaz na prostředek pro továrnu připojení a vlastnost **`res-auth`** je nastavena na hodnotu "Aplikace".	Identifikátor uživatele pro proces aplikačního serveru se tečí do IBM MQ.	Identifikátor uživatele a heslo, které byly předány do metody `ConnectionFactory.createConnection(String username, String password)` , jsou předávány do IBM MQ.
Deskriptor implementace aplikace obsahuje odkaz na prostředek pro továrnu připojení a vlastnost **`res-auth`** je nastavena na hodnotu "Kontejner".	Identifikátor uživatele a heslo zadané v aliasu ověřování pro faktorii připojení jsou odtékávány do produktu IBM MQ.	Identifikátor uživatele a heslo zadané v aliasu ověřování pro faktorii připojení jsou odtékávány do produktu IBM MQ.
Deskriptor implementace aplikace obsahuje odkaz na prostředek pro továrnu připojení, která má vlastnost **`res-auth`** nastavenou na hodnotu "Kontejner", a aplikace byla nakonfigurována s aliasem ověřování.	Identifikátor uživatele a heslo zadané v aliasu pro ověření, které aplikace bylo konfigurováno pro použití, teklo až do IBM MQ.	Identifikátor uživatele a heslo zadané v aliasu pro ověření, které aplikace bylo konfigurováno pro použití, teklo až do IBM MQ.

Tabulka 71. Režim CLIENT

Konfigurace	**Volání aplikace `ConnectionFactory.createConnection()`**	**Volání aplikace `ConnectionFactory.createConnection(String username, String password)`**
Deskriptor implementace aplikace neobsahuje odkaz na prostředek pro továrnu připojení a továrna připojení je nakonfigurována pro použití kanálu produktu IBM MQ , jehož vlastnost MCAUSER není nastavena na hodnotu unset.	Identifikátor uživatele pro proces aplikačního serveru se tečí do IBM MQ.	Identifikátor uživatele a heslo, které byly předány do metody `ConnectionFactory.createConnection(String username, String password)` , jsou předávány do IBM MQ.

Tabulka 71. Režim CLIENT (pokračování)		
Konfigurace	**Volání aplikace `ConnectionFactory.createConnection()`**	**Volání aplikace `ConnectionFactory.createConnection(String username, String password)`**
Deskriptor implementace aplikace neobsahuje odkaz na prostředek pro továrnu připojení a továrna připojení je konfigurována pro použití kanálu produktu IBM MQ , který má vlastnost MCAUSER nastavenou na identifikátor uživatele.	Identifikátor uživatele určený vlastností MCAUSER na kanálu produktu IBM MQ , který je konfigurován pro použití továrny připojení, se tečí do IBM MQ.	Identifikátor uživatele určený vlastností MCAUSER na kanálu produktu IBM MQ , který je konfigurován pro použití továrny připojení, se tečí do IBM MQ.
Deskriptor implementace aplikace obsahuje odkaz na prostředek pro továrnu připojení, která má vlastnost **`res-auth`** je nastavena na hodnotu *Aplikace* a továrna připojení je konfigurována tak, aby používala kanál produktu IBM MQ , jehož vlastnost MCAUSER má nenastavenou vlastnost.	Identifikátor uživatele pro proces aplikačního serveru se tečí do IBM MQ.	Identifikátor uživatele a heslo, které byly předány do metody `ConnectionFactory.createConnection(String username, String password)` , jsou předávány do IBM MQ.
Deskriptor implementace aplikace obsahuje odkaz na prostředek pro továrnu připojení, která má vlastnost **`res-auth`** je nastavena na hodnotu *Aplikace* a továrna připojení je nakonfigurována pro použití kanálu produktu IBM MQ , který má vlastnost MCAUSER nastavenou na identifikátor uživatele.	Identifikátor uživatele, který je určen vlastností MCAUSER na kanálu produktu IBM MQ , který je konfigurován pro použití továrny připojení, se tečí do IBM MQ.	Identifikátor uživatele, který je určen vlastností MCAUSER na kanálu produktu IBM MQ , který je konfigurován pro použití továrny připojení, se tečí do IBM MQ.
Deskriptor implementace aplikace obsahuje odkaz na prostředek pro továrnu připojení, která má vlastnost **`res-auth`** je nastavena na hodnotu "*Kontejner* a továrna připojení je konfigurována pro použití kanálu produktu IBM MQ , který má vlastnost MCAUSER, jejíž nastavení není zrušeno.	Identifikátor uživatele a heslo zadané v aliasu ověřování pro faktorii připojení jsou odtékávány do produktu IBM MQ.	Identifikátor uživatele a heslo zadané v aliasu ověřování pro faktorii připojení jsou odtékávány do produktu IBM MQ.

Tabulka 71. Režim CLIENT (pokračování)		
Konfigurace	**Volání aplikace `ConnectionFactory.createConnection()`**	**Volání aplikace `ConnectionFactory.createConnection(String username, String password)`**
Deskriptor implementace aplikace obsahuje odkaz na prostředek pro továrnu připojení, která má vlastnost **`res-auth`** je nastavena na hodnotu "*Kontejner* a továrna připojení je konfigurována pro použití kanálu produktu IBM MQ , který má vlastnost MCAUSER nastavenou na identifikátor uživatele.	Identifikátor uživatele, který je určen vlastností MCAUSER na kanálu produktu IBM MQ , který je konfigurován pro použití továrny připojení, se tečí do IBM MQ.	Identifikátor uživatele, který je určen vlastností MCAUSER na kanálu produktu IBM MQ , který je konfigurován pro použití továrny připojení, se tečí do IBM MQ.
Deskriptor implementace aplikace obsahuje odkaz na prostředek pro továrnu připojení, která má vlastnost **`res-auth`** je nastavena na hodnotu "*Kontejner* , a aplikace byla nakonfigurována s aliasem ověřování a továrna připojení je nakonfigurována pro použití kanálu produktu IBM MQ , který má vlastnost MCAUSER, nenastavena	Identifikátor uživatele a heslo zadané v aliasu pro ověření, které aplikace bylo konfigurováno pro použití, teklo až do IBM MQ.	Identifikátor uživatele a heslo zadané v aliasu pro ověření, které aplikace bylo konfigurováno pro použití, teklo až do IBM MQ.
Deskriptor implementace aplikace obsahuje odkaz na prostředek pro továrnu připojení, která má vlastnost **`res-auth`** , je nastavena na hodnotu *Kontejner* a aplikace byla nakonfigurována s aliasem ověřování a továrna připojení je konfigurována pro použití kanálu produktu IBM MQ , který má sadu MCAUSER nastavenou na identifikátor uživatele.	Identifikátor uživatele, který je určen vlastností MCAUSER na kanálu produktu IBM MQ , který je konfigurován pro použití továrny připojení, se tečí do IBM MQ.	Identifikátor uživatele, který je určen vlastností MCAUSER na kanálu produktu IBM MQ , který je konfigurován pro použití továrny připojení, se tečí do IBM MQ.

Použití továrny připojení pomocí přímého vyhledávání

Po definování továrny připojení poskytovatele systému zpráv produktu IBM MQ může podniková aplikace vyhledat definici továrny připojení a použít ji k vytvoření připojení produktu JMS ke správci front produktu IBM MQ . To lze provést prostřednictvím přímého vyhledávání.

Chcete-li použít přímé vyhledávání, podniková aplikace se připojí k úložišti produktu JNDI aplikačního serveru, a to tak, že provede následující volání metody:

```
InitialContext ctx = new InitialContext();
```

Po připojení k úložišti JNDI podniková aplikace poté identifikuje definici továrny připojení s použitím názvu produktu JNDI továrny připojení takto:

```
ConnectionFactory cf = (ConnectionFactory) ctx.lookup("jms/myCF");
```

Notes:

- Vývojář aplikací musí při vývoji podnikové aplikace znát název produktu JNDI pro požadovanou továrnu připojení. Vzhledem k tomu, že název JNDI je pevně naprogramován uvnitř aplikace, pokud se změní název produktu JNDI , je třeba aplikaci znovu zapsat a znovu implementovat.
- Je-li použita definice továrny připojení tímto způsobem, jméno uživatele a heslo zadané v aliasu pro ověření (které továrna připojení bylo nakonfigurováno k použití), se netečou do IBM MQ. Účelem je zabránit neautorizovaným aplikacím v identifikaci továrny připojení a jeho použití pro připojení k zabezpečeným systémům IBM MQ .

Jméno uživatele a heslo, které jsou přenášena do IBM MQ , závisí na metodě, která se používá k vytvoření připojení JMS z továrny připojení.

Pokud aplikace vytvoří připojení JMS pomocí této metody:

```
ConnectionFactory.createConnection()
```

výchozí identita uživatele je předána do produktu IBM MQ. Jedná se o jméno uživatele a heslo, které spustilo aplikační server, na kterém je spuštěna podniková aplikace.

Případně může aplikace vytvořit připojení JMS voláním metody:

```
ConnectionFactory.createConnection(String username, String password)
```

Pokud aplikace provedla přímý pohled na továrnu připojení a pak tuto metodu vyvolala, jméno uživatele a heslo, které byly předány do metody `createConnection()` , se tečí do IBM MQ.

Důležité: Před verzí IBM MQ 8.0produkt IBM MQ zpracoval kontrolu autorizace, pouze aby se ujistil, že jméno uživatele, které bylo přenášeno dolů, má oprávnění pro přístup ke správci front.

Na heslo nebyly provedeny žádné kontroly. Chcete-li provést kontrolu ověření a ověřit, že jméno uživatele a heslo byly platné, musí být zapsána uživatelská procedura zabezpečení kanálu produktu IBM MQ . Podrobnosti o tom, jak to lze provést, najdete v tématu Uživatelské programy zabezpečení kanálu.

V produktu IBM MQ 8.0správce front kontroluje heslo spolu se jménem uživatele.

Použití továrny připojení prostřednictvím nepřímého vyhledávání

Pokud zapisujete podnikovou aplikaci, pokud je název produktu JNDI továrny připojení neznámý, nebo pokud má být aplikace nainstalována na různé aplikační servery pomocí jiné továrny připojení, s jiným názvem produktu JNDI (v závislosti na tom, jaký aplikační server je nainstalován na), lze továrnu připojení vyhledat pomocí odkazu na prostředek. To lze provést prostřednictvím nepřímého vyhledávání.

Příklad

Místo přímého vyhledávání továrny připojení pomocí produktu `jms/myCF`obsahuje podniková aplikace odkaz na prostředek, který má lokální název JNDI : `jms/myResourceReferenceCF`.

Chcete-li použít tento název produktu JNDI , připojí se aplikace k úložišti produktu JNDI na aplikačním serveru stejným způsobem, jako kdyby aplikace prováděla přímé vyhledávání:

```
InitialContext ctx = new InitialContext();
```

Místo toho, aby aplikace přímo neidentifikoval `jms/myCF` , identifikuje aplikace JNDI název odkazu na prostředek:

```
ConnectionFactory cf = (ConnectionFactory) ctx.lookup("java:comp/env/jms/
myResourceReferenceCF");
```

Chcete-li sdělit aplikačnímu serveru, že podniková aplikace provádí nepřímý vzhled, musíte zadat předponu `java:comp/env` pro lokální název produktu JNDI .

Když je aplikace implementována, uživatel mapuje JNDI název odkazu na prostředek `jms/myResourceReferenceCF` na název rozhraní JNDI továrny připojení, kterou aplikace již vytvořila: `jms/myCF`.

Při spuštění aplikace vyhledá továrnu připojení produktu JMS pomocí lokálního názvu JNDI , který aplikační server mapuje na: `jms/myCF`. Tuto továrnu připojení pak aplikace použije k vytvoření připojení k produktu IBM MQ.

Aliasy ověřování a nepřímé vyhledávání

Odkaz na prostředek dále umožňuje definovat další vlastnosti, které mění chování poskytované továrny připojení. Jedna z vlastností odkazu na prostředek je **res-auth**. Hodnota této vlastnosti určuje, zda má podniková aplikace používat alias ověřování továrny připojení, na kterou odkazuje odkaz na prostředek při vytváření připojení k produktu IBM MQ (pokud byl definován alias ověřování), nebo pokud aplikace určuje své vlastní jméno uživatele a heslo.

Výchozí hodnota této vlastnosti je *Aplikace*. To znamená, že jméno uživatele a heslo, které jsou přenášena do správce front, je-li vytvořeno připojení JMS , je určeno samotnou aplikací. Alias ověřování továrny připojení, na který se mapy odkazů prostředku nepoužívá.

Aplikace mohou vytvářet připojení JMS pomocí jedné z následujících metod:

- `ConnectionFactory.createConnection()`
- `ConnectionFactory.createConnection(String username, String password)`

Pokud aplikace používá produkt `ConnectionFactory.createConnection()` a produkt **res-auth** je nastaven na hodnotu *Aplikace*, je výchozí identita uživatele přenášena do IBM MQ. Jedná se o jméno uživatele a heslo, které spustilo aplikační server, na kterém je spuštěna podniková aplikace.

Pokud aplikace používá produkt `ConnectionFactory.createConnection(String username, String password)` a produkt **res-auth** je nastaven na hodnotu *Aplikace*, jméno uživatele a heslo předané této metodě jsou odeslány do produktu IBM MQ.

Chcete-li použít alias ověřování definovaný v továrně připojení, na kterou odkazuje mapování prostředků při vytváření připojení, je třeba nastavit vlastnost **res-auth** na hodnotu *Kontejner*. Když aplikace vytvoří připojení JMS , použijí se podrobnosti aliasu ověřování, i když volání `createConnection` uvádí jméno uživatele a heslo.

Přepsání aliasu ověřování při použití nepřímého vyhledávání

Pokud aplikace používá odkaz na prostředek, který má vlastnost **res-auth** nastavenou na hodnotu *Kontejner*, můžete potlačit alias ověřování, které se používá při vytváření připojení produktu JMS .

Chcete-li potlačit alias ověřování, musí odkaz na prostředek obsahovat nadbytečnou vlastnost s názvem **authDataAlias**, která je mapována na existující alias ověřování, který již byl vytvořen v prostředí aplikačního serveru, do kterého bude aplikace implementována. Tuto vlastnost můžete zadat u všech odkazů na prostředky, které byly vytvořeny pomocí nástrojů produktu Rational , které poskytuje produkt IBM.

Při použití této metody můžete při použití továrny připojení produktu JMS , která byla nepřímo vyhledána, použít jiný alias ověřování. Pokud zadaný alias ověřování neexistuje, lze po instalaci podnikové aplikace zadat nový. Další informace naleznete v tématu *Odkazy na prostředky* v dokumentaci produktu WebSphere Application Server .

Související informace pro WebSphere Application Server 8.5.5
Odkazy na prostředek
Související informace pro WebSphere Application Server 8.0
Odkazy na prostředek
Související informace pro WebSphere Application Server 7.0
Odkazy na prostředek

Vyrovnávání pracovní zátěže pro objekty bean řízené zprávami při použití klastrů produktu WebSphere Application Server

Při použití aplikací typu message-driven bean implementovaných v klastru produktu WebSphere Application Server 7.0 a 8.0 a konfigurování pro spuštění v běžném režimu poskytovatele systému zpráv produktu IBM WebSphere MQ se jeden z členů klastru zpracovává většinu zpráv. Pracovní zátěž členů klastru můžete vyvážit tak, aby bylo možné distribuovat zpracování zpráv mezi více než jednoho člena klastru.

Produkt IBM WebSphere MQ 7.0 zavedl novou funkci s názvem **Asynchronous consume**, která umožňuje aplikacím spotřebovávat zprávy asynchronně z fronty pomocí rozhraní API s názvem **MQCB** a **MQCTL**.

Aplikace typu message-driven bean spuštěné uvnitř WebSphere Application Server 7.0 a 8.0, které používají normální režim poskytovatele systému zpráv produktu IBM WebSphere MQ , budou automaticky využívat tuto funkci. Když se aplikace spustí, nastaví na místo určení JMS asynchronní spotřebitele, které byly nakonfigurovány pro monitorování voláním **MQCB**. Rozhraní API **MQCTL** se pak volá, aby označilo, že aplikace je připravena přijímat zprávy z cíle JMS .

Pokud byly aplikace typu message-driven bean implementovány do klastru produktu WebSphere Application Server , každý člen klastru nastaví asynchronního spotřebitele pro místo určení JMS , že objekt typu message-driven bean je monitorován pro zprávy. Správce front produktu IBM WebSphere MQ 7.0 , který je hostitelem cíle JMS , je poté odpovědný za upozornění člena klastru, je-li k dispozici vhodná zpráva pro místo určení JMS .

Před IBM WebSphere MQ 7.0.1 Fix Pack 6budou správci front upřednostňovat prvního člena klastru, aby nastavili svého asynchronního odběratele na cíli JMS. Tento člen klastru bude první, který má být upozorněn, když přijde vhodná zpráva na místo určení JMS . Následně první člen klastru pro spuštění aplikace objektu typu message-driven bean bude zpracovávat většinu vhodných zpráv, které přicházejí do cíle JMS .

Když se produkt WebSphere Application Server připojuje ke správci front produktu IBM WebSphere MQ 7.0.1 Fix Pack 6 nebo novější, budou zprávy, které dorazí do cíle JMS , distribuovány rovnoměrněji na všechny asynchronní spotřebitele, které byly registrovány v místě určení JMS . V případě aplikací typu message-driven bean implementovaných uvnitř klastru WebSphere Application Server 7.0 a 8.0 to znamená, že zprávy budou distribuovány rovnoměrněji mezi členy klastru.

Související informace
Konfigurace vlastnosti PROVIDERVERSION

Použití balíku záhlaví produktu IBM MQ

Balík záhlaví IBM MQ poskytuje sadu nápovědných rozhraní a tříd, které můžete použít k manipulaci se záhlavími IBM MQ zprávy. Obvykle se používá balík záhlaví produktu IBM MQ , protože chcete provádět administrativní služby pomocí příkazového serveru (pomocí zpráv PCF (Programmable Command Format)).

Informace o této úloze

Balík záhlaví produktu IBM MQ se nachází v balících `com.ibm.mq.headers` a `com.ibm.mq.pcf` . Tuto poskytovanou službu lze použít pro obě dvě alternativní rozhraní API, která IBM MQ poskytuje pro použití v aplikacích produktu Java :

- IBM MQ classes for Java (také označováno jako IBM MQ Base Java).
- IBM MQ classes for Java Message Service (IBM MQ classes for JMS, také označovaný jako IBM MQ JMS).

IBM MQ Základní aplikace produktu Java obvykle manipulují s objekty MQMessage a třídy podpory záhlaví mohou s těmito objekty přímo komunikovat, protože nativně rozumí základním rozhraním produktu IBM MQ Java .

V produktu IBM MQ JMSje informačním obsahem zprávy zpravidla řetězec nebo objekt bajtového pole, kterému lze manipulovat s proudy DataInput a DataOutput . Balík záhlaví produktu IBM MQ lze použít pro interakci s těmito datovými proudy a je vhodný pro manipulaci se zprávami produktu MQ , které jsou odesílány a přijímány aplikacemi produktu IBM MQ JMS .

Therefore, although the IBM MQ Headers package contains references to the IBM MQ Base Java package, it is also intended for use within IBM MQ JMS applications and is suitable for use within Java Platform, Enterprise Edition (Java EE) environments.

Typickým způsobem, jak můžete použít balík záhlaví produktu IBM MQ , je manipulovat s administračními zprávami ve formátu PCF (Programmable Command Format), například z následujících důvodů:

- Pro přístup k podrobnostem o prostředku IBM MQ .
- Chcete-li monitorovat hloubku fronty.
- Zablokování přístupu k frontě.

Při použití zpráv PCF s rozhraním API produktu IBM MQ JMS lze tento způsob administrace prostředků zaměřených na aplikaci provádět z aplikací produktu Java EE , aniž by bylo nutné použít rozhraní API produktu IBM MQ Base Java .

Procedura

- Chcete-li použít balík záhlaví produktu IBM MQ k manipulaci se záhlavími zpráv pro produkt IBM MQ classes for Java, prohlédněte si téma “Použití s IBM MQ classes for Java” na stránce 476.
- Chcete-li použít balík záhlaví produktu IBM MQ k manipulaci se záhlavími zpráv pro produkt IBM MQ classes for JMS, prohlédněte si téma “Použití s IBM MQ classes for JMS” na stránce 477.

Použití s IBM MQ classes for Java

Aplikace produktu IBM MQ classes for Java obvykle manipulují s objekty MQMessage a třídy podpory záhlaví mohou s těmito objekty přímo komunikovat, protože nativně rozumí rozhraním produktu IBM MQ classes for Java .

Informace o této úloze

Produkt IBM MQ poskytuje ukázkové aplikace, které demonstrují použití balíku záhlaví produktu IBM MQ se základním Java rozhraním API produktu IBM MQ (IBM MQ classes for Java).

Ukázky zobrazují dvě věci:

- Jak vytvořit zprávu PCF pro provedení administrativní akce a analýzu zprávy odezvy.
- Jak odeslat tuto zprávu PCF pomocí produktu IBM MQ classes for Java.

V závislosti na použité platformě se tyto ukázky nainstalují do adresáře `pcf` v adresáři `samples` nebo `tools` instalace produktu IBM MQ (viz “Instalační adresáře produktu IBM MQ classes for Java” na stránce 315).

Postup

1. Vytvořte zprávu PCF, abyste provedli administrativní akci a analyzoval zprávu odpovědi.
2. Pošleme tuto zprávu PCF pomocí IBM MQ classes for Java.

Související pojmy

“Zpracování záhlaví zpráv produktu IBM MQ pomocí produktu IBM MQ classes for Java” na stránce 341
Poskytují se třídy Java představující různé typy záhlaví zprávy. Poskytují se také dvě pomocné třídy.

“Práce se zprávami PCF s IBM MQ classes for Java” na stránce 346
Třídy Java jsou poskytovány pro vytváření a analýzu zpráv strukturovaných PCF a pro usnadnění odesílání požadavků PCF a shromažďování odpovědí PCF.

Použití s IBM MQ classes for JMS

Chcete-li použít záhlaví IBM MQ se serverem IBM MQ classes for JMS, budete provádět stejné základní kroky jako u produktu IBM MQ classes for Java. Zprávu PCF lze vytvořit a odezvu analyzovat přesně stejným způsobem pomocí balíku záhlaví produktu IBM MQ a stejného vzorového kódu jako u produktu IBM MQ classes for Java.

Informace o této úloze

Chcete-li odeslat zprávu PCF pomocí rozhraní API produktu IBM MQ , musí být informační obsah zprávy zapsán do zprávy JMS bajtů a odeslán pomocí standardních rozhraní API produktu JMS . Jediná úvaha je, že zpráva nesmí obsahovat záhlaví JMS RFH2 ani žádná jiná záhlaví se specifickými hodnotami v MQMD.

Chcete-li odeslat zprávu PCF, proveďte následující kroky. Způsob vytvoření zprávy PCF a informace extrahované ze zprávy odpovědi jsou stejné jako u produktu IBM MQ classes for Java (viz “Použití s IBM MQ classes for Java” na stránce 476).

Postup

1. Vytvořte místo určení fronty JMS , které představuje SYSTEM.ADMIN.COMMAND.QUEUE.

   Aplikace produktu IBM MQ JMS odesílají zprávy PCF do systému SYSTEM.ADMIN.COMMAND.QUEUEa je třeba mít přístup k objektu cíle JMS, který reprezentuje tuto frontu. Cíl musí mít nastaveny následující vlastnosti:

   ```
   WMQ_MQMD_WRITE_ENABLED = YES
   WMQ_MESSAGE_BODY = MQ
   ```

   Pokud používáte produkt WebSphere Application Server, je třeba tyto vlastnosti definovat jako přizpůsobené vlastnosti v místě určení.

   Chcete-li vytvořit místo určení programově z určité aplikace, použijte následující kód:

   ```
   Queue q1 = session.createQueue("SYSTEM.ADMIN.COMMAND.QUEUE");
   ((MQQueue) q1).setIntProperty(WMQConstants.WMQ_MESSAGE_BODY,
       WMQConstants.WMQ_MESSAGE_BODY_MQ);
   ((MQQueue) q1).setMQMDWriteEnabled(true);
   ```

2. Převeďte zprávu PCF na zprávu JMS bajtů obsahující správné hodnoty MQMD.

   Je třeba vytvořit bajtovou zprávu JMS a do ní je zapsána zpráva PCF. Je třeba vytvořit frontu odpovědí, ale tato potřeba nemá žádná specifická nastavení.

   Následující úsek vzorového kódu ukazuje, jak vytvořit zprávu JMS bajtů a zapsat objekt com.ibm.mq.headers, objektpcf.PCFMessage do něj. Objekt PCFMessage (pcfCmd) byl dříve sestaven pomocí balíku záhlaví IBM MQ . (Všimněte si, že balík pro načtení PCFMessage je com.ibm.mq.headers.pcf.PCFMessage).

   ```
    // create the JMS Bytes Message
   final BytesMessage msg = session.createBytesMessage();

   // Create the wrapping streams to put the bytes into the message payload
   ByteArrayOutputStream baos = new ByteArrayOutputStream();
   DataOutput dataOutput = new DataOutputStream(baos);

   // Set the JMSReplyTo so the answer comes back
   msg.setJMSReplyTo(new MQQueue("adminResp"));

   // write the pcf into the stream
   ```

```
pcfCmd.write(dataOutput);
baos.flush();
msg.writeBytes(baos.toByteArray());

// we have taken control of the MD, so need to set all
// flags in the MD that we require - main one is the format
msg.setJMSPriority(4);
msg.setIntProperty(WMQConstants.JMS_IBM_MQMD_PERSISTENCE,
  CMQC.MQPER_NOT_PERSISTENT);
msg.setIntProperty(WMQConstants.JMS_IBM_MQMD_EXPIRY, 300);
msg.setIntProperty(WMQConstants.JMS_IBM_MQMD_REPORT,
    CMQC.MQRO_PASS_CORREL_ID);
msg.setStringProperty(WMQConstants.JMS_IBM_MQMD_FORMAT, "MQADMIN");

// and send the message
sender.send(msg);
```

3. Odešlete zprávu a přijměte odpověď pomocí standardních rozhraní API produktu JMS .
4. Převeďte zprávu odpovědi na zprávu PCF na zpracování.

   Chcete-li načíst zprávu odpovědi a zpracovat ji jako zprávu PCF, použijte následující kód:

```
 // Get the message back
BytesMessage msg = (BytesMessage) consumer.receive();

// get the size of the bytes message & read into an array
int bodySize = (int) msg.getBodyLength();
byte[] data = new byte[bodySize];
msg.readBytes(data);

// Read into Stream and DataInput Stream
ByteArrayInputStream bais = new ByteArrayInputStream(data);
DataInput dataInput = new DataInputStream(bais);

// Pass to PCF Message to process
PCFMessage response = new PCFMessage(dataInput);
```

Související pojmy

“Zprávy produktu JMS” na stránce 121

Zprávy produktu JMS se skládají ze záhlaví, vlastností a těla. JMS definuje pět typů těla zprávy.

IBM i Nastavení IBM MQ na IBM i s Java a JMS

Tato kolekce témat poskytuje přehled o tom, jak jste nastavili a testujete IBM MQ pomocí příkazů Java a JMS v systému IBM i pomocí příkazů jazyka CL nebo prostředí qshell.

Poznámka: Od IBM MQ 8.0, `ldap.jar`, `jndi.jar` a `jta.jar` jsou části sady JDK.

Použití CL příkazů

Hodnota CLASSPATH, kterou jste nastavili, je určena pro testování s bázní jazyka Java MQ , JMS s rozhraním JNDI a JMS bez rozhraní JNDI.

Pokud nepoužíváte soubor `.profile` ve svém adresáři `/home/Userprofile` , budete muset nastavit níže uvedené proměnné prostředí na úrovni systému. Pokud jsou nastaveny pomocí příkazu **WRKENVVAR** , můžete zkontrolovat, zda jsou nastaveny.

1. Chcete-li zobrazit proměnné prostředí pro celý systém, zadejte příkaz: **WRKENVVAR LEVEL (*SYS)**
2. Chcete-li zobrazit proměnné prostředí specifické pro vaši úlohu, zadejte příkaz: **WRKENVVAR LEVEL (*JOB)**
3. Není-li proměnná CLASSPATH nastavena, postupujte takto:

```
ADDENVVAR ENVVAR(CLASSPATH)
       VALUE('.:/QIBM/ProdData/mqm/java/lib/com.ibm.mq.jar
       :/QIBM/ProdData/mqm/java/lib/connector.jar:/QIBM/ProdData/mqm/java/lib
       :/QIBM/ProdData/mqm/java/samples/base
       :/QIBM/ProdData/mqm/java/lib/com.ibm.mqjms.jar
       :/QIBM/ProdData/mqm/java/lib/jms.jar
```

```
        :/QIBM/ProdData/mqm/java/lib/providerutil.jar
        :/QIBM/ProdData/mqm/java/lib/fscontext.jar:') LEVEL(*SYS)
```

4. Není-li proměnná QIBM_MULTI_THREADED nastavena, zadejte následující příkaz:

```
ADDENVVAR ENVVAR(QIBM_MULTI_THREADED) VALUE('Y') LEVEL(*SYS)
```

5. Pokud není soubor QIBM_USE_DESCRIPTOR_STDIO nastaven, zadejte tento příkaz:

```
ADDENVVAR ENVVAR(QIBM_USE_DESCRIPTOR_STDIO) VALUE('I') LEVEL(*SYS)
```

6. Není-li nastavena hodnota QSH_REDIRECTION_TEXTDATA, zadejte následující příkaz:

```
ADDENVVAR ENVVAR(QSH_REDIRECTION_TEXTDATA) VALUE('Y') LEVEL(*SYS)
```

Použití prostředí qshell

Pokud používáte prostředí QSHELL, můžete nastavit prostor `.profile` ve svém adresáři `/home/Userprofile` . Další informace viz dokumentace Qshell Interpreter (qsh).

Uveďte následující informace v `.profile`. Všimněte si, že příkaz CLASSPATH musí být na jednom řádku nebo musí být oddělen na různých řádcích pomocí znaku \ , jak je zobrazeno.

```
CLASSPATH=.:/QIBM/ProdData/mqm/java/lib/com.ibm.mq.jar: \
/QIBM/ProdData/mqm/java/lib/connector.jar: \
/QIBM/ProdData/mqm/java/lib: \
/QIBM/ProdData/mqm/java/samples/base: \
/QIBM/ProdData/mqm/java/lib/com.ibm.mqjms.jar: \
/QIBM/ProdData/mqm/java/lib/jms.jar: \
/QIBM/ProdData/mqm/java/lib/providerutil.jar: \
/QIBM/ProdData/mqm/java/lib/fscontext.jar:
HOME=/home/XXXXX
LOGNAME=XXXXX
PATH=/usr/bin:
QIBM_MULTI_THREADED=Y QIBM_USE_DESCRIPTOR_STDIO=I
QSH_REDIRECTION_TEXTDATA=Y
TERMINAL_TYPE=5250
```

Ujistěte se, že knihovna QMQMJAVA je v seznamu knihoven, vydáním příkazu **DSPLIBL**.

Není-li knihovna QMQMJAVA v seznamu, přidejte ji pomocí následujícího příkazu: **ADDLIBLE LIB (QMQMJAVA)**

IBM i Testování IBM MQ na IBM i s Java

Jak testujete produkt IBM MQ s produktem Java pomocí ukázkového programu MQIVP.

Testování základu IBM MQ Java

Proveďte následující postup:

1. Zadáním následujícího příkazu ověřte, zda je správce front spuštěn a zda je stav správce front ACTIVE.

```
WRKMQM MQMNAME(QMGRNAME)
```

2. Ověřte si to JAVA.CHANNEL byl vytvořen zadáním následujícího příkazu:

```
WRKMQMCHL CHLNAME(JAVA.CHANNEL) CHLTYPE(*SVRCN) MQMNAME(QMGRNAME)
```

   a. Pokud je JAVA.CHANNEL neexistuje, zadejte tento příkaz:

```
CRTMQMCHL CHLNAME(JAVA.CHANNEL) CHLTYPE(*SVRCN) MQMNAME(QMGRNAME)
```

3. Vydáním příkazu **WRKMQMLSR** ověřte, zda je modul listener správce front spuštěn pro port 1414 nebo kterýkoli port, který používáte.
   a. Pokud pro správce front nebyl spuštěn žádný modul listener, zadejte následující příkaz:

```
STRMQMLSR PORT(xxxx) MQMNAME(QMGRNAME)
```

Spuštění ukázkového testovacího programu MQIVP

1. Spusťte qshell, z příkazového řádku zadáním příkazu STRQSH
2. Zadáním příkazu **export** ověřte, že je nastavena správná proměnná CLASSPATH, a pak zadejte příkaz **cd** následujícím způsobem:

```
cd /qibm/proddata/mqm/java/samples/wmqjava/samples
```

3. Spusťte program **java** zadáním následujícího příkazu:

```
java MQIVP
```

Když jste vyzváni k zadání, můžete stisknout klávesu ENTER:

- Typ připojení
- Adresa IP
- Název správce front

a použijte výchozí hodnoty. Tím se ověří vazby produktu, které lze nalézt v knihovně QMQMJAVA.

Výstup se zobrazí podobně jako v následujícím příkladu. Všimněte si, že prohlášení o autorských právech závisí na verzi produktu, který používáte.

```
> java MQIVP
MQSeries for Java Installation Verification Program
5724-H72 (C) Copyright IBM Corp. 2011, 2023. All Rights Reserved.
============================================================

Please enter the IP address of the MQ server :
>
Please enter the queue manager name :
>
Attaching Java program to QIBM/ProdData/mqm/java/lib/connector.JAR.
Success: Connected to queue manager.
Success: Opened SYSTEM.DEFAULT.LOCAL.QUEUE
Success: Put a message to SYSTEM.DEFAULT.LOCAL.QUEUE
Success: Got a message from SYSTEM.DEFAULT.LOCAL.QUEUE
Success: Closed SYSTEM.DEFAULT.LOCAL.QUEUE
Success: Disconnected from queue manager


Tests complete -
SUCCESS: This MQ Transport is functioning correctly.
Press Enter to continue ...
>
$
```

Testování připojení klienta IBM MQ Java

Musíte uvést:

- Typ připojení
- Adresa IP
- Port

- Kanál připojení serveru
- Správce front

Výstup se zobrazí podobně jako v následujícím příkladu. Všimněte si, že prohlášení o autorských právech závisí na verzi produktu, který používáte.

```
> java MQIVP
MQSeries for Java Installation Verification Program
5724-H72 (C) Copyright IBM Corp. 2011, 2023. All Rights Reserved.
============================================================

Please enter the IP address of the MQ server :
> x.xx.xx.xx
Please enter the port to connect to : (1414)
> 1470
Please enter the server connection channel name :
> JAVA.CHANNEL
Please enter the queue manager name :
> KAREN01
Success: Connected to queue manager.
Success: Opened SYSTEM.DEFAULT.LOCAL.QUEUE
Success: Put a message to SYSTEM.DEFAULT.LOCAL.QUEUE
Success: Got a message from SYSTEM.DEFAULT.LOCAL.QUEUE
Success: Closed SYSTEM.DEFAULT.LOCAL.QUEUE
Success: Disconnected from queue manager


Tests complete -
SUCCESS: This MQ Transport is functioning correctly.
Press Enter to continue ...
>
$
```

IBM i Testování IBM MQ na IBM i s JMS

Jak testujete produkt IBM MQ s produktem JMS s rozhraním JNDI a bez rozhraní JNDI

Testování JMS bez rozhraní JNDI pomocí ukázky IVTRun

Proveďte následující postup:

1. Zadáním následujícího příkazu ověřte, zda je správce front spuštěn a zda je stav správce front ACTIVE.

   ```
   WRKMQM MQMNAME(QMGRNAME)
   ```

2. Zadáním příkazu **STRQSH** spusťte qshell, z příkazového řádku.
3. Příkaz **cd** se používá ke změně adresáře následujícím způsobem:

   ```
   cd /qibm/proddata/mqm/java/bin
   ```

4. Spusťte skriptový soubor:

   ```
   IVTRun -nojndi [-m qmgrname]
   ```

Výstup se zobrazí podobně jako v následujícím příkladu. Povšimněte si, že prohlášení o autorských právech závisí na verzích produktů, které používáte:

```
> IVTRun -nojndi -m ELCRTP19

Attaching Java program to
/QIBM/ProdData/mqm/java/lib/com.ibm.mqjms.JAR.
Attaching Java program to
/QIBM/ProdData/mqm/java/lib/jms.JAR.

5724-H72, 5724-B41, 5655-F10 (c) Copyright IBM Corp. 2011, 2023.
All Rights Reserved.
```

```
WebSphere MQ classes for Java(tm) Message Service 5.300
Installation Verification Test


Creating a QueueConnectionFactory
Creating a Connection
Creating a Session
Creating a Queue
Creating a QueueSender
Creating a QueueReceiver
Creating a TextMessage
Sending the message to SYSTEM.DEFAULT.LOCAL.QUEUE
Reading the message back again

Got message:
JMS Message class: jms_text
JMSType: null
JMSDeliveryMode: 2
JMSExpiration: 0
JMSPriority: 4
JMSMessageID: ID:c1d4d840c5d3c3d9e3d7f1f9404040403ccf041f0000c012
JMSTimestamp: 1020273404500
JMSCorrelationID:null
JMSDestination: queue:///SYSTEM.DEFAULT.LOCAL.QUEUE
JMSReplyTo: null
JMSRedelivered: false
JMS_IBM_PutDate:20040326
JMSXAppID:QP0ZSPWT STANLEY 170302
JMS_IBM_Format:MQSTR
JMS_IBM_PutApplType:8
JMS_IBM_MsgType:8
JMSXUserID:STANLEY
JMS_IBM_PutTime:13441354
JMSXDeliveryCount:1
A simple text message from the MQJMSIVT program
Reply string equals original string
Closing QueueReceiver
Closing QueueSender
Closing Session
Closing Connection
IVT completed OK
IVT finished
$
>
$
```

Testování režimu klienta produktu IBM MQ JMS bez rozhraní JNDI

Proveďte následující postup:

1. Zadáním následujícího příkazu ověřte, zda je správce front spuštěn a zda je stav správce front ACTIVE.

   ```
   WRKMQM MQMNAME(QMGRNAME)
   ```

2. Zadáním následujícího příkazu ověřte, zda je vytvořen kanál připojení k serveru:

   ```
   WRKMQMCHL CHLNAME( SYSTEM.DEF.SVRCONN ) CHLTYPE(*SVRCN)
   MQMNAME(QMGRNAME)
   ```

3. Zadáním příkazu **WRKMQMLSR** ověřte, zda je modul listener spuštěn pro správný port.
4. Zadáním příkazu **STRQSH** spusťte qshell, z příkazového řádku.
5. Zadáním příkazu **export** ověřte, že je CLASSPATH správná.
6. Příkaz **cd** se používá ke změně adresáře následujícím způsobem:

   ```
   cd /qibm/proddata/mqm/java/bin
   ```

7. Spusťte skriptový soubor:

   ```
   IVTRun -nojndi -client -m QMgrName -host hostname [-port port] [-channel channel]
   ```

Výstup se zobrazí podobně jako v následujícím příkladu. Všimněte si, že prohlášení o autorských právech závisí na verzích produktů, které používáte.

```
> IVTRun -nojndi -client -m ELCRTP19 -host ELCRTP19 -port 1414 -channel SYSTEM.DEF.SVRCONN

5724-H72, 5724-B41, 5655-F10 (c) Copyright IBM Corp. 2011, 2023.
All Rights Reserved.
WebSphere MQ classes for Java(tm) Message Service 5.300
Installation Verification Test


Creating a QueueConnectionFactory
Creating a Connection
Creating a Session
Creating a Queue
Creating a QueueSender
Creating a QueueReceiver
Creating a TextMessage
Sending the message to SYSTEM.DEFAULT.LOCAL.QUEUE
Reading the message back again

Got message:
JMS Message class: jms_text
JMSType: null
JMSDeliveryMode: 2
JMSExpiration: 0
JMSPriority: 4
JMSMessageID: ID:c1d4d840c5d3c3d9e3d7f1f9404040403ccf041f0000d012
JMSTimestamp: 1020274009970
JMSCorrelationID:null
JMSDestination: queue:///SYSTEM.DEFAULT.LOCAL.QUEUE
JMSReplyTo: null
JMSRedelivered: false
JMS_IBM_PutDate:20040326
JMSXAppID:MQSeries Client for Java
JMS_IBM_Format:MQSTR
JMS_IBM_PutApplType:28
JMS_IBM_MsgType:8
JMSXUserID:QMQM
JMS_IBM_PutTime:14085237
JMSXDeliveryCount:1
A simple text message from the MQJMSIVT program
Reply string equals original string
Closing QueueReceiver
Closing QueueSender
Closing Session
Closing Connection
IVT completed OK
IVT finished
$
```

Testování IBM MQ JMS s rozhraním JNDI

Zadáním následujícího příkazu ověřte, zda je správce front spuštěn a zda je stav správce front ACTIVE.

```
WRKMQM MQMNAME(QMGRNAME)
```

Použití ukázkového testovacího skriptu IVTRun

Proveďte následující postup:

1. Proveďte příslušné změny v souboru `JMSAdmin.config` . Chcete-li upravit tento soubor, použijte příkaz **EDTF** (Upravit soubor) z příkazového řádku IBM i .

   ```
   EDTF '/qibm/proddata/mqm/java/bin/JMSAdmin.config'
   ```

   .

   a. Chcete-li použít protokol LDAP for Weblogic, odeberte tento komentář:

```
INITIAL_CONTEXT_FACTORY=com.sun.jndi.ldap.LdapCtxFactory
```

b. Chcete-li použít protokol LDAP pro produkt WebSphere Application Server, odeberte tento komentář:

```
INITIAL_CONTEXT_FACTORY=com.ibm.ejs.ns.jndi.CNInitialContextFactory
```

c. Chcete-li otestovat systém souborů, odeberte komentář:

```
INITIAL_CONTEXT_FACTORY=com.sun.jndi.fscontext.RefFSContextFactory
```

d. Ujistěte se, že jste vybrali správnou adresu PROVIDER_URL tak, že odeberete komentář z příslušné řádky.

e. Označte jako komentář všechny ostatní řádky pomocí symbolu *#* .

f. Jakmile dokončíte všechny své změny, stiskněte klávesu **F2=Save** a **F3=Exit**.

2. Zadáním příkazu **STRQSH** spusťte qshell, z příkazového řádku.
3. Zadáním příkazu **export** ověřte, že je CLASSPATH správná.
4. Příkaz **cd** se používá ke změně adresáře následujícím způsobem:

```
cd /qibm/proddata/mqm/java/bin
```

5. Spuštěním příkazu **IVTSetup** spusťte skript **IVTSetup** pro vytvoření administrovaných objektů (*MQQueueConnectionFactory* a *MQQueue*).
6. Spusťte skript IVTRun zadáním následujícího příkazu:

```
IVTRun -url providerURL [-icf initCtxFact]
```

Výstup se zobrazí podobně jako v následujícím příkladu. Všimněte si, že prohlášení o autorských právech závisí na verzích produktů, které používáte.

```
> IVTSetup
+ Creating script for object creation within JMSAdmin
+ Calling JMSAdmin in batch mode to create objects
Ignoring unknown flag: -i

5724-H72 (c) Copyright IBM Corp. 2011, 2023. All Rights Reserved.
Starting WebSphere MQ classes for Java(tm) Message Service Administration

InitCtx>
InitCtx>
InitCtx>
InitCtx>
InitCtx>
Stopping MQSeries classes for Java(tm) Message Service Administration

+ Administration done; tidying up files
+ Done!
$

> IVTRun -url file:////tmp/mqjms -icf com.sun.jndi.fscontext.RefFSContextFactory

5724-H72 (c) Copyright IBM Corp. 2011, 2023. All Rights Reserved.
MQSeries classes for Java(tm) Message Service
Installation Verification Test

Using administered objects, please ensure that these are available

Retrieving a QueueConnectionFactory from JNDI
Creating a Connection
Creating a Session
Retrieving a Queue from JNDI
Creating a QueueSender
Creating a QueueReceiver
```

```
Creating a TextMessage
Sending the message to SYSTEM.DEFAULT.LOCAL.QUEUE
Reading the message back again

Got message:
JMS Message class: jms_text
JMSType: null
JMSDeliveryMode: 2
JMSExpiration: 0
JMSPriority: 4
JMSMessageID: ID:c1d4d840c5d3c3d9e3d7f1f9404040403ccf041f0000e012
JMSTimestamp: 1020274903770
JMSCorrelationID:null
JMSDestination: queue:///SYSTEM.DEFAULT.LOCAL.QUEUE
JMSReplyTo: null
JMSRedelivered: false
JMS_IBM_Format:MQSTR
JMS_IBM_PutApplType:8
JMSXDeliveryCount:1
JMS_IBM_MsgType:8
JMSXUserID:STANLEY
JMSXAppID:QP0ZSPWT STANLEY 170308
A simple text message from the MQJMSIVT program
Reply string equals original string
Closing QueueReceiver
Closing QueueSender
Closing Session
Closing Connection
IVT completed OK
IVT finished
$
```

Vývoj aplikací C++

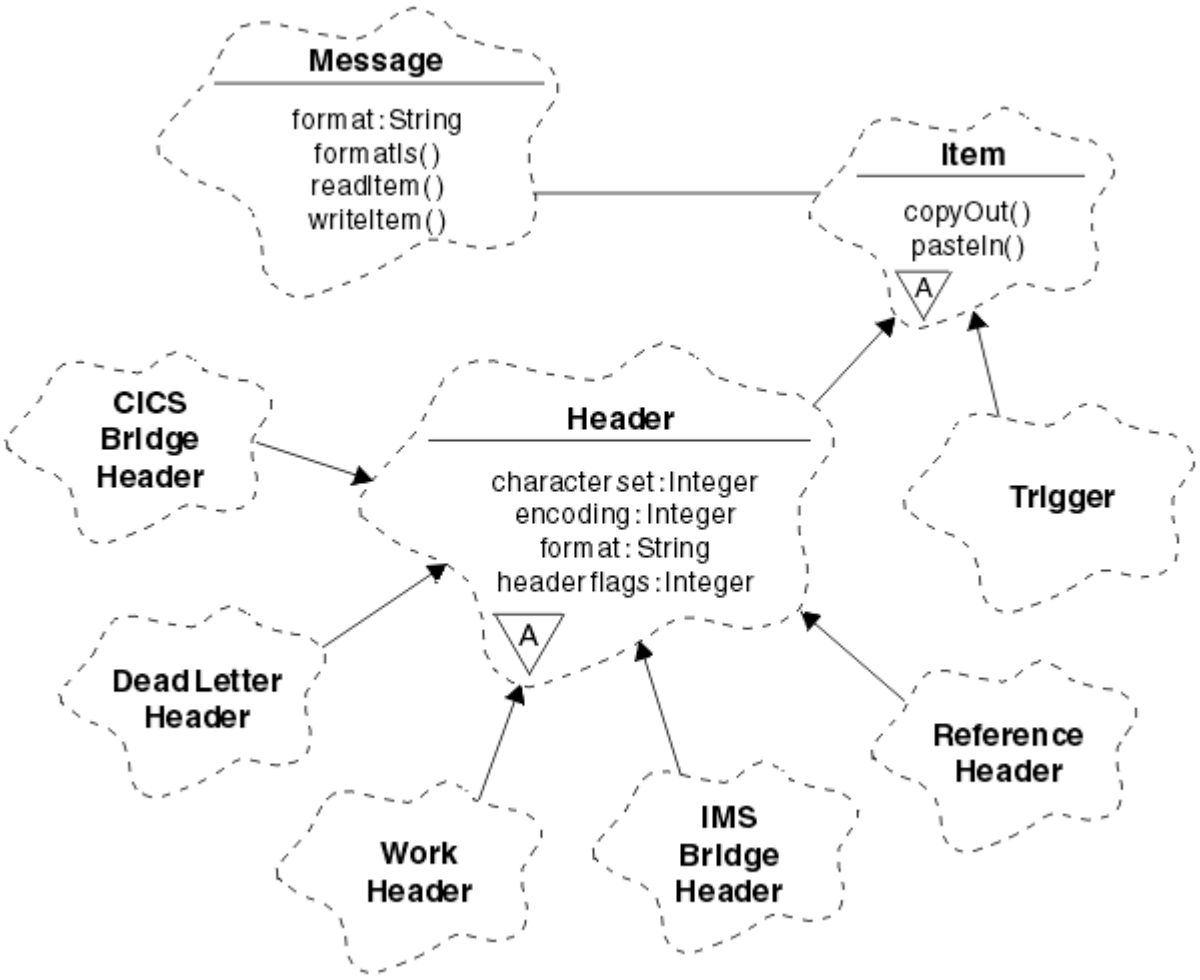
Produkt IBM MQ poskytuje třídy C + + ekvivalentní s objekty IBM MQ a některé další třídy ekvivalentní k datovým typům pole. Poskytuje řadu funkcí, které nejsou prostřednictvím rozhraní MQI k dispozici.

IBM WebSphere MQ 7.0, vylepšení programovacího rozhraní produktu IBM MQ se neaplikují na třídy C + +.

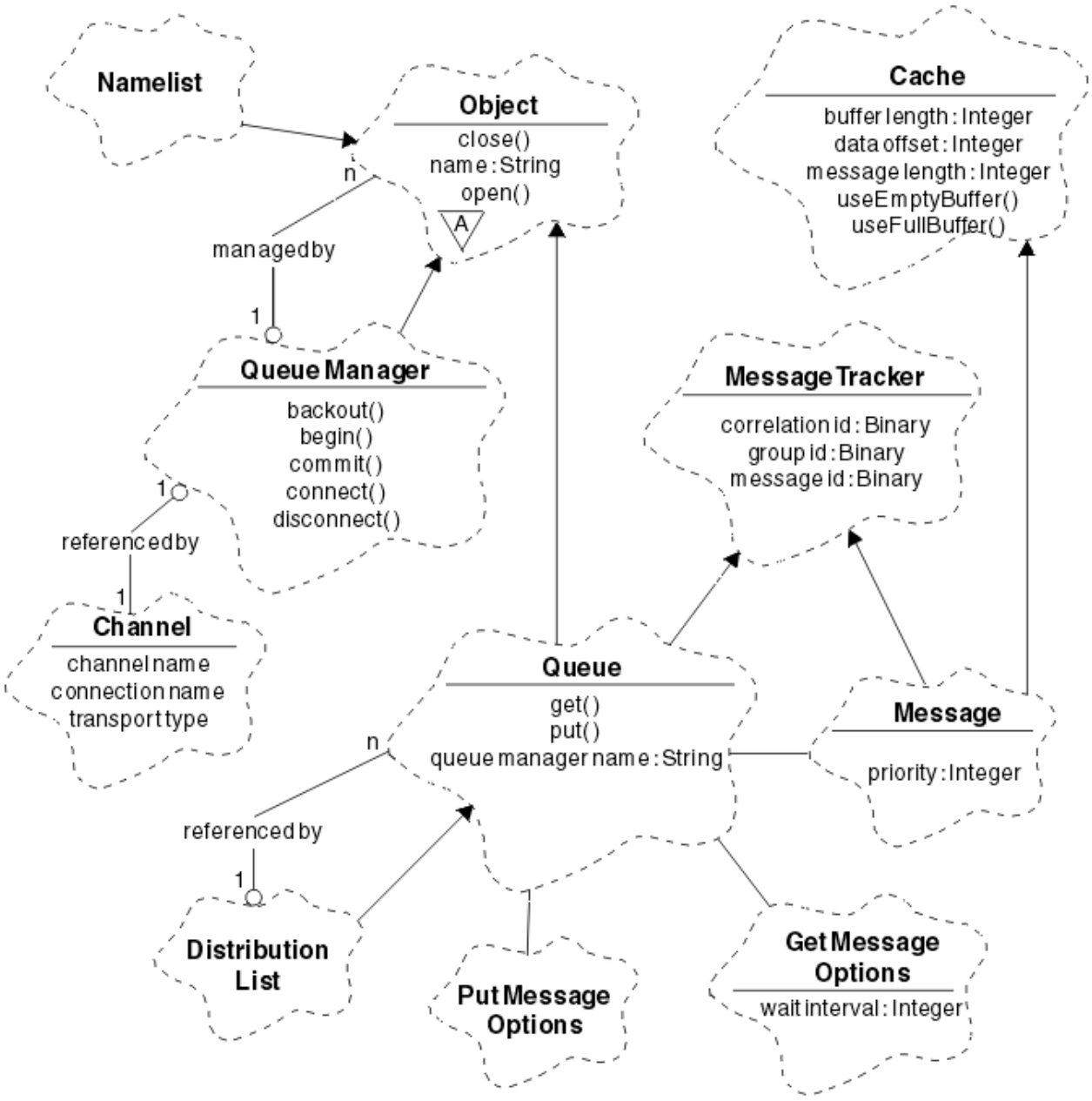
Produkt IBM MQ C++ poskytuje následující funkce:

- Automatická inicializace datových struktur produktu IBM MQ .
- Otevření fronty a otevření fronty správce front s časem ukončení.
- Odpojení implicitního ukončení fronty a správce front.
- Přenos a příjem záhlaví nedoručených zpráv.
- Přenos a příjem záhlaví mostu IMS .
- Referenční přenos a potvrzení záhlaví zprávy.
- Potvrzení přijetí zprávy.
- CICS bridge -přenos a příjem záhlaví.
- Přenos a příjem záhlaví práce.
- Definice kanálu klienta.

Následující záložny třídy Booch ukazují, že všechny třídy jsou obecně rovnoběžné s těmi entitami IBM MQ v procedurálním rozhraní MQI (například pomocí C), které mají buď popisovače, nebo datové struktury. Všechny třídy dědí z třídy ImqError třídy (viz ImqError C++ třída), což umožňuje přiřadit chybový stav ke každému objektu.



Obrázek 56. Třídy C++ produktu IBM MQ (zpracování položek)



Obrázek 57. IBM MQ Třídy C++ (správa front)

Chcete-li správně interpretovat snímky třídy Booch, uvědomte si následující konvence:

- Metody a zajímavé atributy jsou zobrazeny pod názvem *class* .
- Malý trojúhelník v rámci cloudu označuje *abstraktní třídu*.
- *Dědičnost* je označena šipkou pro nadřízenou třídu.
- Nezdobená čára mezi mraky označuje *kooperativní vztah* mezi třídami.
- Čára zdobená číslem označuje *referenční vztah* mezi dvěma třídami. Číslo označuje počet objektů, které se mohou podílet na určitém vztahu najednou.

Následující třídy a datové typy jsou použity v signatur metody C + + u tříd správy front (viz Obrázek 57 na stránce 487) a třídy zpracování položek (viz Obrázek 56 na stránce 486):

- Třída ImqBinary (viz ImqBinary C++ class), která zapouzdřuje bajtová pole, jako například MQBYTE24.
- Datový typ ImqBoolean , který je definován jako **typedef unsigned char ImqBoolean**.
- Třída ImqString (viz ImqString C++ class), která zapouzdřuje znaková pole jako MQCHAR64.

Entity s datovými strukturami jsou v rámci odpovídajících tříd objektů podsunuté. Jednotlivá pole datové struktury (viz Křížový odkaz jazyka C++ a MQI) jsou přístupné pomocí metod.

Objekty s manipulátory jsou dodávány pod hierarchií tříd ImqObject (viz ImqObject C++ class). a poskytněte zapouzdřené rozhraní do MQI. Objekty těchto tříd vykazují inteligentní chování, které může snížit počet vyvolání metody vyžadovaných vzhledem k procedurálnímu rozhraní MQI. Můžete například vytvořit a vyřadit připojení správce front podle potřeby, nebo můžete frontu s odpovídajícími volbami otevřít a zavřít ji.

Třída ImqMessage (viz ImqMessage C++ class). zapouzdřuje datovou strukturu MQMD a slouží také jako zadržovací bod pro uživatelská data a *položky* (viz “Čtení zpráv v C++” na stránce 497). poskytnutím vyrovnávací paměti uložené v mezipaměti. Pro uživatelská data můžete poskytovat vyrovnávací paměti s pevnou délkou a použít ji mnohokrát. Množství dat přítomných ve vyrovnávací paměti se může lišit od jednoho použití k dalšímu. Alternativně může systém poskytovat a spravovat vyrovnávací paměť flexibilní délky. Významné úvahy se stanou velikostí vyrovnávací paměti (dostupné pro příjem zpráv) a skutečně použitou částkou (buď počet bajtů pro přenos, nebo počet skutečně přijatých bajtů).

Související pojmy

“Ukázkové programy C++” na stránce 488
K dispozici jsou čtyři vzorové programy, které demonstrují získávání a vkládání zpráv.

“Pokyny k jazyku C++” na stránce 492
Tato kolekce témat podrobně popisuje aspekty použití jazyka C++ a konvence, které musíte zvážit při zápisu aplikačních programů, které používají rozhraní MQI (Message Queue Interface).

“Příprava dat zprávy v jazyce C++” na stránce 496
Data zprávy jsou připravena ve vyrovnávací paměti, kterou může poskytnout systém nebo aplikace. Pro obě metody existují výhody. Příklady použití vyrovnávací paměti jsou uvedeny.

“Vyvíjení aplikací pro IBM MQ” na stránce 5
Můžete vyvíjet aplikace k odesílání a přijímání zpráv a ke správě správců front a souvisejících prostředků. IBM MQ podporuje aplikace napsané v mnoha různých jazycích a rámcích.

Související odkazy

“Sestavování programů IBM MQ C++” na stránce 503
Adresa URL podporovaných kompilátorů je uvedena spolu s příkazy, které se mají použít ke kompilaci, propojení a spuštění programů C++ a ukázek na platformách IBM MQ .

Související informace

Technický přehled
Křížový odkaz C++ a MQI
IBM MQ Třídy C++

Ukázkové programy C++

K dispozici jsou čtyři vzorové programy, které demonstrují získávání a vkládání zpráv.

Ukázkové programy jsou:

- HELLO WORLD (imqwrld.cpp)
- SPUT (imqsput.cpp)
- SGET (imqsget.cpp)
- DPUT (imqdput.cpp)

Ukázkové programy jsou umístěny v adresářích zobrazených v Tabulka 72 na stránce 489.

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Tabulka 72. Umístění ukázkových programů		
Prostředí	**Adresář obsahující zdroj**	**Adresář obsahující sestavení programy**
AIX	*MQ_INSTALLATION_PATH*/samp	*MQ_INSTALLATION_PATH*/samp/bin/ia
IBM i IBM i	/QIBM/ProdData/mqm/samp/	(viz poznámka “1” na stránce 489)
HP-UX	*MQ_INSTALLATION_PATH*/samp	*MQ_INSTALLATION_PATH*/ samp/bin/ah (viz poznámka “2” na stránce 489)
z/OS z/OS	thlqual.SCSQCPPS	Není
Solaris	*MQ_INSTALLATION_PATH*/samp	*MQ_INSTALLATION_PATH*/ samp/bin/as
Linux	*MQ_INSTALLATION_PATH*/samp	*MQ_INSTALLATION_PATH*/samp/bin/
Windows	*MQ_INSTALLATION_PATH*\tools\cplus\ samples	*MQ_INSTALLATION_PATH*\tools\cplus\ ukázky \bin\vn (viz poznámka “3” na stránce 489)

Notes:

1. Programy sestavené s použitím kompilátoru ILE C + + pro IBM i jsou v knihovně QMQM. Soubory začlenění jsou v /QIBM/ProdData/mqm/inc.
2. Programy sestavené pomocí kompilátoru HP ANSI C + + se nacházejí v adresáři *MQ_INSTALLATION_PATH*/samp/bin/ah. Další informace uvádí téma “Sestavování programů C++ v systému HP-UX” na stránce 504.
3. Programy sestavené pomocí produktu Microsoft Visual Studio se nacházejí v adresáři *MQ_INSTALLATION_PATH*\tools\cplus\samples\bin\vn. Další informace o těchto kompilátorech naleznete v tématu “Sestavování programů C++ v systému Windows” na stránce 510.

Ukázkový program HELLO WORLD (imqwrld.cpp)

Tento ukázkový program C++ ukazuje, jak vložit a získat regulární datagram (strukturu C) pomocí třídy ImqMessage .

Tento program ukazuje, jak vložit a získat regulární datagram (strukturu C) pomocí třídy ImqMessage . Tato ukázka používá málo vyvolání metody, přičemž využívá implicitní vyvolání metod, jako jsou **otevření**, **zavření**a **odpojení**.

Na všech platformách kromě z/OS

Používáte-li připojení k serveru IBM MQ, postupujte podle jedné z následujících procedur:

- Chcete-li použít existující výchozí frontu, SYSTEM.DEFAULT.LOCAL.QUEUE, spusťte program **imqwrlds** bez předání jakýchkoli parametrů
- Chcete-li použít dočasnou dynamicky přiřazenou frontu, spusťte příkaz **imqwrlds** s předáním názvu výchozí modelové fronty SYSTEM.DEFAULT.MODEL.QUEUE.

Používáte-li připojení klienta k produktu IBM MQ, postupujte podle jedné z následujících procedur:

- Nastavte proměnnou prostředí MQSERVER (viz MQSERVER , kde získáte další informace) a spusťte příkaz **imqwrldc**nebo
- Příkaz **imqwrldc** se spustí jako parametry **queue-name**, **queue-manager-name**a **channel-definition**, kde typická hodnota **channel-definition** může být SYSTEM.DEF.SVRCONN/TCP/*název_hostitele* (1414)

zapz/OS

z/OS

Vytvořte a spusťte dávkovou úlohu s použitím ukázkového souboru JCL **imqwrldr**.

Další informace naleznete v dokumentu z/OS Batch, RRS Batch a CICS .

Ukázkový kód

```
extern "C" {
#include <stdio.h>
}

#include <imqi.hpp> // IBM MQ C++

#define EXISTING_QUEUE "SYSTEM.DEFAULT.LOCAL.QUEUE"

#define BUFFER_SIZE 12

static char gpszHello[ BUFFER_SIZE ] = "Hello world" ;
int main ( int argc, char * * argv ) {
  ImqQueueManager manager ;
  int iReturnCode = 0 ;

  // Connect to the queue manager.
  if ( argc > 2 ) {
    manager.setName( argv[ 2 ] );
  }
  if ( manager.connect( ) ) {
    ImqQueue * pqueue = new ImqQueue ;
    ImqMessage * pmsg = new ImqMessage ;

    // Identify the queue which will hold the message.
    pqueue -> setConnectionReference( manager );
    if ( argc > 1 ) {
      pqueue -> setName( argv[ 1 ] );


      // The named queue can be a model queue, which will result in
      // the creation of a temporary dynamic queue, which will be
      // destroyed as soon as it is closed. Therefore we must ensure
      // that such a queue is not automatically closed and reopened.
      // We do this by setting open options which will avoid the need
      // for closure and reopening.
      pqueue -> setOpenOptions( MQOO_OUTPUT | MQOO_INPUT_SHARED |
                                MQOO_INQUIRE );
    } else {
      pqueue -> setName( EXISTING_QUEUE );

      // The existing queue is not a model queue, and will not be
      // destroyed by automatic closure and reopening. Therefore we
      // will let the open options be selected on an as-needed basis.
      // The queue will be opened implicitly with an output option
      // during the "put", and then implicitly closed and reopened
      // with the addition of an input option during the "get".
    }

    // Prepare a message containing the text "Hello world".
    pmsg -> useFullBuffer( gpszHello , BUFFER_SIZE );
```

```
    pmsg -> setFormat( MQFMT_STRING );

    // Place the message on the queue, using default put message
    // Options.
    // The queue will be automatically opened with an output option.
    if ( pqueue -> put( * pmsg ) ) {
      ImqString strQueue( pqueue -> name( ) );

      // Discover the name of the queue manager.
      ImqString strQueueManagerName( manager.name( ) );
      printf( "The queue manager name is %s.\n",
              (char *)strQueueManagerName );

      // Show the name of the queue.
      printf( "Message sent to %s.\n", (char *)strQueue );

      // Retrieve the data message just sent ("Hello world" expected)
      // from the queue, using default get message options. The queue
      // is automatically closed and reopened with an input option
      // if it is not already open with an input option. We get the
      // message just sent, rather than any other message on the
      // queue, because the "put" will have set the ID of the message
      // so, as we are using the same message object, the message ID
      // acts as in the message object, a filter which says that we
      // are interested in a message only if it has this
      // particular ID.

      if ( pqueue -> get( * pmsg ) ) {
        int iDataLength = pmsg -> dataLength( );

        // Show the text of the received message.
        printf( "Message of length %d received, ", iDataLength );

        if ( pmsg -> formatIs( MQFMT_STRING ) ) {
          char * pszText = pmsg -> bufferPointer( );

          // If the last character of data is a null, then we can
          // assume that the data can be interpreted as a text
          // string.
          if ( ! pszText[ iDataLength - 1 ] ) {
            printf( "text is \"%s\".\n", pszText );
          } else {
            printf( "no text.\n" );
          }

        } else {
          printf( "non-text message.\n" );
        }
      } else {
        printf( "ImqQueue::get failed with reason code %ld\n",
                pqueue -> reasonCode( ) );
        iReturnCode = (int)pqueue -> reasonCode( );
      }

    } else {
      printf( "ImqQueue::open/put failed with reason code %ld\n",
              pqueue -> reasonCode( ) );
      iReturnCode = (int)pqueue -> reasonCode( );
    }

    // Deletion of the queue will ensure that it is closed.
    // If the queue is dynamic then it will also be destroyed.
    delete pqueue ;
    delete pmsg ;

  } else {
    printf( "ImqQueueManager::connect failed with reason code %ld\n"
            manager.reasonCode( ) );
    iReturnCode = (int)manager.reasonCode( );
  }

  // Destruction of the queue manager ensures that it is
  // disconnected.  If the queue object were still available
  // and open (which it is not), the queue would be closed
  // prior to disconnection.

  return iReturnCode ;
}
```

Vzorové programy SPUT (imqsput.cpp) a SGET (imqsget.cpp)

Tyto programy C++ umisťují zprávy do pojmenované fronty a načítají je z fronty.

Tyto ukázky zobrazují použití následujících tříd:

- ImqError (viz ImqError Třída C++)
- ImqMessage (viz ImqMessage C++ class)
- ImqObject (viz ImqObject C++ class)
- ImqQueue (viz ImqQueue C++ class)
- Správce ImqQueueManager (viz ImqQueueManager C++ class)

Postupujte podle příslušných pokynů pro spouštění programů.

Na všech platformách kromě z/OS

1. Spusťte příkaz **imqsputs** *název-fronty*.
2. Zadejte řádky textu na konzole. Tyto řádky jsou umístěny jako zprávy do zadané fronty.
3. Zadejte řádek null, abyste ukončili vstup.
4. Spuštěním příkazu **imqsgets** *název-fronty* načtete všechny řádky a zobrazíte je na konzole.

z/OS Další informace viz “Sestavování programů jazyka C++ v produktu z/OS Batch, Dávka RRS a CICS” na stránce 512.

zapz/OS

z/OS

1. Vytvořte a spusťte dávkovou úlohu s použitím ukázkového souboru JCL **imqsputr**. Zprávy se čtou z datové sady SYSIN.
2. Vytvořte a spusťte dávkovou úlohu s použitím ukázkového souboru JCL **imqsgetr**. Zprávy se načtou z fronty a odešlou do datové sady SYSPRINT.

Ukázkový program DPUT (imqdput.cpp)

Tento ukázkový program C++ umisťuje zprávy do rozdělovníku, který se skládá ze dvou front.

Funkce DPUT zobrazuje použití třídy ImqDistributionList (viz ImqDistributionList C++ class). Tato ukázka není v produktu z/OSpodporována.

1. Spuštěním příkazu **imqdputs** *queue-name-1 queue-name-2* umístěte zprávy do dvou pojmenovaných front.
2. Spuštěním příkazu **imqsgets** *queue-name-1* a **imqsgets** *queue-name-2* načtěte zprávy z těchto front.

Pokyny k jazyku C++

Tato kolekce témat podrobně popisuje aspekty použití jazyka C++ a konvence, které musíte zvážit při zápisu aplikačních programů, které používají rozhraní MQI (Message Queue Interface).

Soubory záhlaví C++

Soubory záhlaví jsou poskytovány jako součást definice rozhraní MQI, které vám pomohou při zápisu aplikačních programů IBM MQ v jazyce C + +.

Tyto soubory záhlaví jsou shrnuty v následující tabulce.

Tabulka 73. Soubory záhlaví C/C++	
Název souboru	**Obsah**
IMQI.HPP	Třídy rozhraní MQI jazyka C++ (včetně CMQC.H a IMQTYPE.H)
IMQTYPE.H	Definuje datový typ **ImqBoolean** .
CMQC.H	Struktury dat MQI a konstanty souboru typu manifest

Chcete-li zlepšit přenositelnost aplikací, uveďte název souboru záhlaví malými písmeny na direktivě preprocesoru **#include** :

```
#include <imqi.hpp> // C++ classes
```

Metody a atributy jazyka C++

Názvy metod jsou ve smíšeném případě. Na parametry a návratové hodnoty se vztahují různé aspekty. Atributy jsou přístupné pomocí metody set a get, jak je to vhodné.

Parametry metod, které jsou *const* , jsou určeny pouze pro vstup. Parametry s podpisy včetně ukazatele (*) nebo odkazu (&) jsou předávány odkazem. Návratové hodnoty, které nezahrnují ukazatel nebo odkaz, jsou předávány hodnotou; v případě vrácených objektů se jedná o nové entity, které se staly odpovědností volajícího.

Některé signatury metod obsahují položky, které berou výchozí hodnotu, pokud nejsou uvedeny. Takové položky jsou vždy na konci podpisů a jsou označeny rovnítkem (=); hodnota za rovnítkem označuje výchozí hodnotu, která se použije, pokud je položka vynechána.

Všechny názvy metod v těchto třídách jsou malými písmeny, počínaje malými písmeny. Každé slovo, kromě prvního v názvu metody, začíná velkým písmenem. Zkratky se nepoužijí, pokud jejich význam není všeobecně srozumitelná. Použité zkratky zahrnují *id* (pro identitu) a *sync* (pro synchronizaci).

K atributům objektu se přistupuje pomocí metody set a get. Metoda set začíná slovem *set* ; metoda get nemá žádnou předponu. Je-li atribut *jen pro čtení*, není nastavena žádná metoda set.

Atributy jsou inicializovány na platné stavy během konstrukce objektu a stav objektu je vždy konzistentní.

Datové typy v C++

Všechny datové typy jsou definovány příkazem C **typedef** .

Typ **ImqBoolean** je definován jako **unsigned char** v IMQTYPE.H a mohou mít hodnoty TRUE a FALSE. Můžete použít objekty třídy **ImqBinary** místo polí **MQBYTE** a objekty třídy **ImqString** na místě **znak ***. Mnoho metod vrací objekty místo znaků **char** nebo **MQBYTE** k usnadnění správy ukládání dat. Všechny návratové hodnoty se stanou odpovědností volajícího a v případě vráceného objektu může být úložiště likvidován pomocí odstranění.

Manipulace s binárními řetězci v jazyce C++

Řetězce binárních dat jsou deklarovány jako objekty třídy **ImqBinary** . Objekty této třídy lze kopírovat, porovnat a nastavit pomocí známých operátorů C. Vzorový kód je poskytnut.

Následující ukázka kódu zobrazuje operace na binárním řetězci:

```
#include <imqi.hpp> // C++ classes

ImqMessage message ;
ImqBinary id, correlationId ;
MQBYTE24 byteId ;

correlationId.set( byteId, sizeof( byteId ) ); // Set.
id = message.id( );                            // Assign.
```

```
if ( correlationId == id ) {                    // Compare.
...
```

Manipulace se znakovými řetězci v C++

Znaková data jsou často vrácena v objektech třídy **ImqString** , které lze přetypovat na **char *** pomocí operátoru převodu. Třída ImqString obsahuje metody pomáhající při zpracování znakových řetězců.

Jsou-li znaková data přijata nebo vrácena pomocí metod rozhraní MQI C + +, jsou znaková data vždy null-ukončena a mohou mít libovolnou délku. Avšak určité limity jsou ukládány produktem IBM MQ , které mohou vést k oříznutí informací. Za účelem usnadnění správy ukládání dat se často vrací znaková data do objektů třídy **ImqString** . Tyto objekty lze přetypovat na **char *** pomocí operátoru převodu, který je poskytován, a používá se pro účely *jen pro čtení* v mnoha situacích, kdy je vyžadován znak **char *** .

Poznámka: Výsledek převodu **char *** z objektu třídy **ImqString** může mít hodnotu null.

Ačkoli lze funkce jazyka C použít na **char ***, existují speciální metody třídy **ImqString** , které jsou vhodnější; **operátor length** () je ekvivalentní s **strlen** a **storage** () označuje velikost paměti přidělené pro znaková data.

Initial state of objects in C++

Všechny objekty mají konzistentní počáteční stav, který odráží jejich atributy. Počáteční hodnoty jsou definovány v popisech tříd.

Použití jazyka C z jazyka C++

Používáte-li funkce C z programu C + +, zahrňte příslušná záhlaví.

Následující příklad ukazuje program `string.h` zahrnutý v programu C + +:

```
extern "C" {
#include <string.h>
}
```

Notační konvence C + +

Tento příklad ukazuje, jak vyvolat metody a deklarovat parametry.

Tato ukázka kódu používá metody a parametry **ImqBoolean ImqQueue**:: **get** (**ImqMessage &** *msg*).

Deklarujte a použijte parametry následujícím způsobem:

```
ImqQueueManager * pmanager ;    // Queue manager
ImqQueue * pqueue ;             // Message queue
ImqMessage msg ;                // Message
char szBuffer[ 100 ];          // Buffer for message data

pmanager = new ImqQueueManager ;
pqueue = new ImqQueue ;
pqueue -> setName( "myreplyq" );
pqueue -> setConnectionReference( pmanager );

msg.useEmptyBuffer( szBuffer, sizeof( szBuffer ) );

if ( pqueue -> get( msg ) ) {
  long lDataLength = msg.dataLength( );

...
}
```

Implicitní operace v C++

Několik operací může nastat implicitně, *právě včas*, aby splnil podmínky splnění předpokladů pro úspěšné provedení metody. Tyto implicitní operace jsou connect, open, reopen, close a disconnect. Můžete řídit připojení a otevřít implicitní chování pomocí atributů třídy.

Připojit

Objekt správce ImqQueueje připojen automaticky pro libovolnou metodu, která vede k volání MQI (viz Křížový odkaz jazyka C++ a MQI).

Otevřené

Objekt ImqObject je automaticky otevřen pro libovolnou metodu, která vede k volání MQGET, MQINQ, MQPUT nebo MQSET. Použijte metodu **openFor** k uvedení jedné nebo více relevantních hodnot **volby otevření** .

Znovu otevřít

Objekt ImqObject je automaticky znovu otevřen pro libovolnou metodu, jejímž výsledkem je volání MQGET, MQINQ, MQPUT nebo MQSET, kde je objekt již otevřen, ale existující **volby otevření** nejsou dostatečné k tomu, aby bylo volání MQI úspěšné. Objekt je dočasně zavřen pomocí dočasné hodnoty **close options** MQCO_NONE. Použijte metodu **openFor** k přidání relevantního**volba otevření**.

Opětné otevření může způsobit problémy za určitých okolností:

- Dočasná dynamická fronta je zničena, když je uzavřena a nelze ji nikdy znovu otevřít.
- Fronta otevřená pro výlučný vstup (ať už explicitně, nebo při výchozím nastavení) může být přístupná pro jiné osoby v okně příležitost během zavírání a opětovného otevření.
- Poloha kurzoru při procházení se ztratí, když je fronta zavřena. Tato situace nebrání zavření a opětovnému otevření, ale zabrání následnému použití kurzoru, dokud nebude znovu použit MQGMO_BROTE_FIRST.
- Kontext poslední načtené zprávy je ztracen, když je fronta uzavřena.

Pokud se některý z těchto okolností vyskytne nebo může být předvídaný, vyhněte se opětovnému otevření explicitním nastavením vhodných **voleb otevření** před otevřením objektu (ať už explicitně, nebo implicitně).

Nastavení **voleb otevření** explicitně pro složité situace obsluhující fronty vede k lepšímu výkonu a vyhýbá se problémům spojeným s opětným otevřením.

Zavřít

Objekt ImqObject je automaticky zavřen v libovolném okamžiku, kdy stav objektu již není životaschopný, například pokud je odkaz na připojení ImqObject přerušen, nebo je-li objekt ImqObject zničen.

Odpojit

Správce ImqQueueje automaticky odpojen v libovolném okamžiku, kdy připojení již není realizovatelné, například pokud je odkaz na připojení ImqObject přerušený, nebo pokud je objekt ImqQueueManager zničen.

Binární a znakové řetězce v jazyce C++

Třída ImqString zapouzdřuje tradiční datový formát *char ** . Třída ImqBinary zapouzdřuje binární bajtové pole. Některé metody, které nastavují znaková data, by mohly data oříznout.

Metody, které nastavují znak (**char ***) data vždy berou kopii dat, ale některé metody mohou tuto kopii oříznout, protože určité limity jsou ukládány produktem IBM MQ.

Třída ImqString (viz ImqString C++ class). zapouzdřuje tradiční **znak *** a poskytuje podporu pro:

- Porovnání
- Zřetězení
- Kopírování
- převod typu Integer-to-text a text-to-integer
- Extrakce tokenu (slova)
- Překlad velkých písmen

Třída ImqBinary (viz ImqBinary C++ class). zapouzdřuje binární bajtová pole libovolné velikosti. Zejména se používá k uchování následujících atributů:

- **accounting token (účtovací token)** (MQBYTE32)
- **connection tag = příznak připojení** (MQBYTE128)
- **correlation id** (MQBYTE24).
- **token zařízení** (MQBYTE8)
- **group id (ID skupiny)** (MQBYTE24)
- **instance id** (MQBYTE24)
- **message id** (MQBYTE24)
- **token zprávy** (MQBYTE16)
- **ID instance transakce** (MQBYTE16)

Kde tyto atributy patří do objektů v následujících třídách:

- ImqCICSBridgeHeader (viz ImqCICSBridgeHeader C++ class)
- ImqGetMessageOptions (viz ImqGetMessageOptions C++ class).
- ImqIMSBridgeHeader (viz ImqIMSBridgeHeader C++ class)
- ImqMessageTracker (viz ImqMessageTracker C++ class)
- Správce ImqQueueManager (viz ImqQueueManager C++ class)
- záhlaví ImqReference(viz ImqReferenceTřída C++ záhlaví)
- Záhlaví ImqWork(viz ImqWorkHeader C++ class)

Třída ImqBinary také poskytuje podporu pro porovnání a kopírování.

Nepodporované funkce v jazyce C++

Třídy a metody jazyka C++ produktu IBM MQ jsou nezávislé na platformě IBM MQ . Mohou tedy nabízet některé funkce, které nejsou na určitých platformách podporovány.

If you try to use a function on a platform on which it is not supported, the function is detected by IBM MQ but not by the C++ language bindings. Produkt IBM MQ oznámí chybu vašemu programu, stejně jako jakákoli jiná chyba MQI.

Systém zpráv v C++

Tato kolekce témat obsahuje podrobné informace o přípravě, čtení a zápisu zpráv v jazyce C + +.

Příprava dat zprávy v jazyce C++

Data zprávy jsou připravena ve vyrovnávací paměti, kterou může poskytnout systém nebo aplikace. Pro obě metody existují výhody. Příklady použití vyrovnávací paměti jsou uvedeny.

Při odeslání zprávy se data zprávy nejprve připraví ve vyrovnávací paměti spravované objektem ImqCache (viz ImqCache C++ class). Vyrovnávací paměť je přidružena (podle dědičnosti) k jednotlivým objektům ImqMessage (viz ImqMessage C++ class): může být dodána aplikací (buď pomocí metody **useEmptyBuffer** , nebo **useFullBuffer**) nebo automaticky systémem. Výhoda aplikace dodávající vyrovnávací paměť zpráv znamená, že v mnoha případech není nutné kopírovat žádné datové kopie,

protože aplikace může přímo používat připravené datové oblasti. Nevýhodou je, že zadaná vyrovnávací paměť má pevnou délku.

Vyrovnávací paměť lze znovu použít a počet přenesených bajtů lze každou dobu měnit, a to pomocí metody **setMessageLength** před přenosem.

Je-li systém určen automaticky, je počet dostupných bajtů spravován systémem a data lze zkopírovat do vyrovnávací paměti zprávy například pomocí metody ImqCache **write** nebo metody ImqMessage **writeItem** . Velikost vyrovnávací paměti zpráv roste podle potřeby. Jak velikost vyrovnávací paměti narůstá, nedošlo ke ztrátě dříve zapsaných dat. Velká nebo vícedílná zpráva může být zapsána v sekvenčních kouskách.

Následující příklady ukazují zjednodušené odeslání zprávy.

1. Použít připravená data ve vyrovnávací paměti dodané uživatelem

```
char szBuffer[ ] = "Hello world" ;

msg.useFullBuffer( szBuffer, sizeof( szBuffer ) );
msg.setFormat( MQFMT_STRING );
```

2. Použijte připravená data v uživatelem zadané vyrovnávací paměti, kde velikost vyrovnávací paměti překračuje velikost dat.

```
char szBuffer[ 24 ] = "Hello world" ;

msg.useEmptyBuffer( szBuffer, sizeof( szBuffer ) );
msg.setFormat( MQFMT_STRING );
msg.setMessageLength( 12 );
```

3. Kopírovat data do vyrovnávací paměti zadané uživatelem

```
char szBuffer[ 12 ];

msg.useEmptyBuffer( szBuffer, sizeof( szBuffer ) );
msg.setFormat( MQFMT_STRING );
msg.write( 12, "Hello world" );
```

4. Kopírovat data do vyrovnávací paměti dodávané systémem

```
msg.setFormat( MQFMT_STRING );
msg.write( 12, "Hello world" );
```

5. Kopírovat data do vyrovnávací paměti dodávané systémem s použitím objektů (objekty nastavují formát zprávy a obsah)

```
ImqString strText( "Hello world" );

msg.writeItem( strText );
```

Čtení zpráv v C++

Vyrovnávací paměť může být dodána aplikací nebo systémem. K datům lze přistupovat přímo z vyrovnávací paměti nebo číst postupně. Pro každý typ zprávy existuje třída ekvivalentní. Ukázkový kód je uveden.

Při příjmu dat může aplikace nebo systém dodat vhodnou vyrovnávací paměť zpráv. Stejnou vyrovnávací paměť lze použít pro vícenásobný přenos i pro více příjemku pro konkrétní objekt ImqMessage . Je-li vyrovnávací paměť zpráv dodána automaticky, bude zvětšeno, aby bylo možné přijmout jakoukoli délku dat, která je k dispozici. Avšak vyrovnávací paměť zpráv dodaná aplikací nemusí být dostatečně velká, aby zadržela přijatá data. Potom může dojít k oříznutí nebo selhání, v závislosti na volbách použitých pro přijetí zprávy.

K příchozím datům lze přistupovat přímo z vyrovnávací paměti zpráv, v takovém případě délka dat označuje celkové množství příchozích dat. Jinou možností je, že příchozí data lze číst sekvenčně z vyrovnávací paměti zpráv. V tomto případě se ukazatel na data adresuje dalšímu bajtu příchozích dat a datový ukazatel a délka dat se aktualizují pokaždé, když se čtou data.

Položky jsou části zprávy, vše v uživatelské oblasti vyrovnávací paměti zpráv, které je třeba zpracovávat postupně a samostatně. Kromě běžných uživatelských dat může být položka záhlavím se smrtícím písmenem nebo zprávou spouštěče. Položky jsou vždy přidruženy s formáty zpráv; formáty zpráv **nejsou** vždy přidruženy k položkám.

Pro každou položku, která odpovídá poznatelném formátu zprávy produktu IBM MQ , existuje třída objektu. Pro záhlaví a jednu zprávu spouštěče se nachází jedna záhlaví a jedna záhlaví. Pro uživatelská data neexistuje žádná třída objektu. To znamená, že jakmile budou rozpoznatelné formáty vyčerpány, bude zbývající část zpracování ponechána aplikačnímu programu. Třídy pro data uživatelů lze zapisovat prostřednictvím specializace třídy ImqItem .

Následující příklad zobrazuje příjemku zprávy, která bere v úvahu celou řadu potenciálních položek, které mohou předcházet datům uživatele, v imaginární situaci. Uživatelská data, která nejsou položkou, jsou definována jako vše, co nastane za položkami, které lze identifikovat. Automatická vyrovnávací paměť (výchozí) se používá k uložení libovolného množství dat zprávy.

```
ImqQueue queue ;
ImqMessage msg ;

if ( queue.get( msg ) ) {

  /* Process all items of data in the message buffer. */
  do while ( msg.dataLength( ) ) {
    ImqBoolean bFormatKnown = FALSE ;
    /* There remains unprocessed data in the message buffer. */

    /* Determine what kind of item is next. */

    if ( msg.formatIs( MQFMT_DEAD_LETTER_HEADER ) ) {
      ImqDeadLetterHeader header ;
      /* The next item is a dead-letter header.                */
      /* For the next statement to work and return TRUE,       */
      /* the correct class of object pointer must be supplied. */
      bFormatKnown = TRUE ;

      if ( msg.readItem( header ) ) {
        /* The dead-letter header has been extricated from the */
        /* buffer and transformed into a dead-letter object.   */
        /* The encoding and character set of the dead-letter   */
        /* object itself are MQENC_NATIVE and MQCCSI_Q_MGR.    */
        /* The encoding and character set from the dead-letter */
        /* header have been copied to the message attributes   */
        /* to reflect any remaining data in the buffer.        */

        /* Process the information in the dead-letter object.  */
        /* Note that the encoding and character set have       */
        /* already been processed.                             */
        ...
      }
      /* There might be another item after this, */
      /* or just the user data.                  */
    }
    if ( msg.formatIs( MQFMT_TRIGGER ) ) {
      ImqTrigger trigger ;
      /* The next item is a trigger message.                   */
      /* For the next statement to work and return TRUE,       */
      /* the correct class of object pointer must be supplied. */
      bFormatKnown = TRUE ;
      if ( msg.readItem( trigger ) ) {

        /* The trigger message has been extricated from the */
        /* buffer and transformed into a trigger object.    */
        /* Process the information in the trigger object. */
        ...
      }

      /* There is usually nothing after a trigger message. */
    }
```

```
    if ( msg.formatIs( FMT_USERCLASS ) ) {
      UserClass object ;
      /* The next item is an item of a user-defined class.     */
      /* For the next statement to work and return TRUE,       */
      /* the correct class of object pointer must be supplied. */
      bFormatKnown = TRUE ;

      if ( msg.readItem( object ) ) {
        /* The user-defined data has been extricated from the */
        /* buffer and transformed into a user-defined object. */

        /* Process the information in the user-defined object. */
        ...
      }

      /* Continue looking for further items. */
    }
    if ( ! bFormatKnown ) {
      /* There remains data that is not associated with a specific*/
      /* item class.                                               */
      char * pszDataPointer = msg.dataPointer( );       /* Address.*/
      int iDataLength = msg.dataLength( );              /* Length. */

      /* The encoding and character set for the remaining data are */
      /* reflected in the attributes of the message object, even   */
      /* if a dead-letter header was present.                      */
      ...

    }

  }
}
```

V tomto příkladě je FMT_USERCLASS konstantou představující 8znakový název formátu přidružený k objektu třídy `UserClassa` je definován aplikací.

`UserClass` je odvozen z třídy ImqItem (viz ImqItem C++ class) a implementuje metody **copyOut** a **pasteIn** z této třídy.

Následující dva příklady zobrazují kód ze třídy ImqDeadLetterHeader (viz ImqDeadLetterHeader C++ class). První příklad ukazuje vlastní zapouzdřenou zprávu- *writing* kód.

```
// Insert a dead-letter header.
// Return TRUE if successful.
ImqBoolean ImqDeadLetterHeader :: copyOut ( ImqMessage & msg ) {
  ImqBoolean bSuccess ;
  if ( msg.moreBytes( sizeof( omqdlh ) ) ) {
    ImqCache cacheData( msg ); // Preserve original message content.
    // Note original message attributes in the dead-letter header.
    setEncoding( msg.encoding( ) );
    setCharacterSet( msg.characterSet( ) );
    setFormat( msg.format( ) );

    // Set the message attributes to reflect the dead-letter header.
    msg.setEncoding( MQENC_NATIVE );
    msg.setCharacterSet( MQCCSI_Q_MGR );
    msg.setFormat( MQFMT_DEAD_LETTER_HEADER );
    // Replace the existing data with the dead-letter header.
    msg.clearMessage( );
    if ( msg.write( sizeof( omqdlh ), (char *) & omqdlh ) ) {
      // Append the original message data.
      bSuccess = msg.write( cacheData.messageLength( ),
                            cacheData.bufferPointer( ) );
    } else {
      bSuccess = FALSE ;
    }
  } else {
    bSuccess = FALSE ;
  }
  // Reflect and cache error in this object.
  if ( ! bSuccess ) {
    setReasonCode( msg.reasonCode( ) );
    setCompletionCode( msg.completionCode( ) );
  }

  return bSuccess ;
}
```

Druhý příklad ukazuje vlastní zapouzdřenou zprávu- *reading* kód.

```
// Read a dead-letter header.
// Return TRUE if successful.
ImqBoolean ImqDeadLetterHeader :: pasteIn ( ImqMessage & msg ) {
  ImqBoolean bSuccess = FALSE ;

  // First check that the eye-catcher is correct.
  // This is also our guarantee that the "character set" is correct.
  if ( ImqItem::structureIdIs( MQDLH_STRUC_ID, msg ) ) {
    // Next check that the "encoding" is correct, as the MQDLH
    // contains numeric data.
    if ( msg.encoding( ) == MQENC_NATIVE ) {

      // Finally check that the "format" is correct.
      if ( msg.formatIs( MQFMT_DEAD_LETTER_HEADER ) ) {
        char * pszBuffer = (char *) & omqdlh ;
        // Transfer the MQDLH from the message and move pointer on.
        if ( bSuccess = msg.read( sizeof( omdlh ), pszBuffer ) ) {
          // Update the encoding, character set and format of the
          // message to reflect the remaining data.
          msg.setEncoding( encoding( ) );
          msg.setCharacterSet( characterSet( ) );
          msg.setFormat( format( ) );
        } else {

          // Reflect the cache error in this object.
          setReasonCode( msg.reasonCode( ) );
          setCompletionCode( msg.completionCode( ) );
        }
      } else {
        setReasonCode( MQRC_INCONSISTENT_FORMAT );
        setCompletionCode( MQCC_FAILED );
      }
    } else {
      setReasonCode( MQRC_ENCODING_ERROR );
      setCompletionCode( MQCC_FAILED );
    {
  } else {
    setReasonCode( MQRC_STRUC_ID_ERROR );
    setCompletionCode( MQCC_FAILED );
  }

  return bSuccess ;
}
```

S automatickým vyrovnávacími paměťmi je vyrovnávací paměť *nestálá*. To znamená, že data vyrovnávací paměti se mohou nacházet v jiném fyzickém umístění po každém vyvolání metody **get** . Proto jsou při každém odkazování na data vyrovnávací paměti použita metoda **bufferPointer** nebo **dataPointer** k přístupu k datům zpráv.

Můžete chtít, aby program vyjmula pevnou oblast pro příjem dat zprávy. V takovém případě vyvolejte metodu **useEmptyBuffer** předtím, než použijete metodu **get** .

Použití pevné a neautomatické oblasti omezuje zprávy na maximální velikost, takže je důležité vzít v úvahu volbu MQGMO_ACCEPT_TRUNCATED_MSG objektu ImqGetMessageOptions . Není-li tato volba zadána (výchozí nastavení), lze očekávat kód příčiny MQRC_TRUNCATED_MSG_FAILED. Je-li zadána tato volba, může být v závislosti na návrhu aplikace očekáván kód příčiny MQRC_TRUNCATED_MSG_ACCEPTED.

Následující příklad ukazuje, jak lze použít pevnou oblast paměti pro příjem zpráv:

```
char * pszBuffer = new char[ 100 ];

msg.useEmptyBuffer( pszBuffer, 100 );
gmo.setOptions( MQGMO_ACCEPT_TRUNCATED_MSG );
queue.get( msg, gmo );

delete [ ] pszBuffer ;
```

V tomto fragmentu kódu lze vyrovnávací paměť vždy adresovat přímo, pomocí *pszBuffer*, na rozdíl od použití metody **bufferPointer** . Je však lepší použít metodu **dataPointer** pro přístup generálního

účelu. Aplikace (nikoli objekt třídy ImqCache) musí vyřadit uživatelsky definovanou (neautomatizovanou) vyrovnávací paměť.

Upozornění: Určení ukazatele Null a nulové délky s parametrem **useEmptyBuffer** neurčí pevnou délku vyrovnávací paměti o délce nula, jak by se dalo očekávat. Tato kombinace je interpretována jako požadavek na ignorování jakékoli předchozí vyrovnávací paměti definované uživatelem, a namísto toho se vrátí k použití automatické vyrovnávací paměti.

Zápis zprávy do fronty dead-letter v jazyce C++

Příklad kódu programu pro zápis zprávy do fronty nedoručených zpráv.

Typickým případem zprávy s více částmi je jedna z nich obsahující záhlaví s dead-letter. Data ze zprávy, která nelze zpracovat, se připojí k záhlaví s dead-letter.

```
ImqQueueManager mgr ;        // The queue manager.
ImqQueue queueIn ;           // Incoming message queue.
ImqQueue queueDead ;         // Dead-letter message queue.
ImqMessage msg ;             // Incoming and outgoing message.
ImqDeadLetterHeader header ; // Dead-letter header information.

// Retrieve the message to be rerouted.
queueIn.setConnectionReference( mgr );
queueIn.setName( MY_QUEUE );
queueIn.get( msg );

// Set up the dead-letter header information.
header.setDestinationQueueManagerName( mgr.name( ) );
header.setDestinationQueueName( queueIn.name( ) );
header.setPutApplicationName( /* ? */ );
header.setPutApplicationType( /* ? */ );
header.setPutDate( /* TODAY */ );
header.setPutTime( /* NOW */ );
header.setDeadLetterReasonCode( FB_APPL_ERROR_1234 );

// Insert the dead-letter header information. This will vary
// the encoding, character set and format of the message.
// Message data is moved along, past the header.
msg.writeItem( header );

// Send the message to the dead-letter queue.
queueDead.setConnectionReference( mgr );
queueDead.setName( mgr.deadLetterQueueName( ) );
queueDead.put( msg );
```

Zápis zprávy do mostu IMS v jazyce C++

Vzorový kód programu pro zápis zprávy na most IMS .

Zprávy odeslané do mostu IBM MQ - IMS mohou používat speciální záhlaví. Hlavička mostu IMS má jako předponu předponu pravidelných dat zprávy.

```
ImqQueueManager    mgr;         // The queue manager.
ImqQueue           queueBridge; // IMS bridge message queue.
ImqMessage         msg;         // Outgoing message.
ImqIMSBridgeHeader header;      // IMS bridge header.

// Set up the message.
//
// Here we are constructing a message with format
// MQFMT_IMS_VAR_STRING, and appropriate data.
//
msg.write( 2,       /* ? */ );         // Total message length.
msg.write( 2,       /* ? */ );         // IMS flags.
msg.write( 7,       /* ? */ );         // Transaction code.
msg.write( /* ? */, /* ? */ );         // String data.
msg.setFormat( MQFMT_IMS_VAR_STRING ); // The format attribute.

// Set up the IMS bridge header information.
//
// The reply-to-format is often specified.
// Other attributes can be specified, but all have default values.
//
```

```
header.setReplyToFormat( /* ? */ );

// Insert the IMS bridge header into the message.
//
// This will:
// 1) Insert the header into the message buffer, before the existing
//    data.
// 2) Copy attributes out of the message descriptor into the header,
//    for example the IMS bridge header format attribute will now
//    be set to MQFMT_IMS_VAR_STRING.
// 3) Set up the message attributes to describe the header, in
//    particular setting the message format to MQFMT_IMS.
//
msg.writeItem( header );

// Send the message to the IMS bridge queue.
//
queueBridge.setConnectionReference( mgr );
queueBridge.setName( /* ? */ );
queueBridge.put( msg );
```

Zápis zprávy do produktu CICS bridge v jazyce C++

Vzorový kód programu pro zápis zprávy do CICS bridge.

Zprávy odeslané do produktu IBM MQ for z/OS pomocí produktu CICS bridge vyžadují zvláštní záhlaví. Záhlaví CICS bridge je vloženo do běžných dat zprávy.

```
ImqQueueManager mgr ;         // The queue manager.
ImqQueue queueIn ;            // Incoming message queue.
ImqQueue queueBridge ;        // CICS bridge message queue.
ImqMessage msg ;              // Incoming and outgoing message.
ImqCicsBridgeHeader header ;  // CICS bridge header information.

// Retrieve the message to be forwarded.
queueIn.setConnectionReference( mgr );
queueIn.setName( MY_QUEUE );
queueIn.get( msg );

// Set up the CICS bridge header information.
// The reply-to format is often specified.
// Other attributes can be specified, but all have default values.
header.setReplyToFormat( /* ? */ );

// Insert the CICS bridge header information.  This will vary
// the encoding, character set and format of the message.
// Message data is moved along, past the header.
msg.writeItem( header );

// Send the message to the CICS bridge queue.
queueBridge.setConnectionReference( mgr );
queueBridge.setName( /* ? */ );
queueBridge.put( msg );
```

Psaní zprávy se záhlavím pro práci v jazyce C++

Vzorový kód programu pro zápis zprávy určené pro frontu spravovanou produktem z/OS Workload Manager.

Zprávy odeslané do produktu IBM MQ for z/OS, které jsou určeny pro frontu spravovanou produktem z/OS Workload Manager, vyžadují speciální záhlaví. Hlavička práce je předřazeno k regulárním datům zprávy.

```
ImqQueueManager mgr ;             // The queue manager.
ImqQueue queueIn ;                // Incoming message queue.
ImqQueue queueWLM ;               // WLM managed queue.
ImqMessage msg ;                  // Incoming and outgoing message.
ImqWorkHeader header ;            // Work header information

// Retrieve the message to be forwarded.
queueIn.setConnectionReference( mgr );
queueIn.setName( MY_QUEUE );
queueIn.get( msg );

// Insert the Work header information.  This will vary
```

```
// the encoding, character set and format of the message.
// Message data is moved along, past the header.
msg.writeItem( header );

// Send the message to the WLM managed queue.
queueWLM.setConnectionReference( mgr );
queueWLM.setName( /* ? */ );
queueWLM.put( msg );
```

Sestavování programů IBM MQ C++

Adresa URL podporovaných kompilátorů je uvedena spolu s příkazy, které se mají použít ke kompilaci, propojení a spuštění programů C++ a ukázek na platformách IBM MQ .

Seznam kompilátorů pro každou podporovanou platformu a verzi produktu IBM MQnaleznete v tématu Systémové požadavky pro IBM MQ.

Příkaz, který potřebujete zkompilovat a propojit váš program IBM MQ C + +, závisí na vaší instalaci a požadavcích. Příklady, které následují, zobrazují typické kompilační a propojené příkazy pro některé kompilátory s použitím výchozí instalace produktu IBM MQ na řadě platforem.

Sestavování programů C++ v systému AIX

Sestavujte programy IBM MQ C++ v produktu AIX pomocí kompilátoru XL C Enterprise Edition .

Klient

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

32bitová aplikace bez podprocesů

```
xlC -o imqsputc_32 imqsput.cpp -qchars=signed -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -limqc23ia -limqb23ia -lmqic
```

32bitovou aplikaci s podprocesy

```
xlC_r -o imqsputc_32_r imqsput.cpp -qchars=signed -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -limqc23ia_r -limqb23ia_r -lmqic_r
```

64bitová aplikace bez podprocesů

```
xlC -q64 -o imqsputc_64 imqsput.cpp -qchars=signed -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -limqc23ia -limqb23ia -lmqic
```

Aplikace s 64bitovým podprocesem

```
xlC_r -q64 -o imqsputc_64_r imqsput.cpp -qchars=signed -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -limqc23ia_r -limqb23ia_r -lmqic_r
```

Server

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

32bitová aplikace bez podprocesů

```
xlC -o imqsput_32 imqsput.cpp -qchars=signed -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -limqs23ia -limqb23ia -lmqm
```

32bitovou aplikaci s podprocesy

```
xlC_r -o imqsput_32_r imqsput.cpp -qchars=signed -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -limqs23ia_r -limqb23ia_r -lmqm_r
```

64bitová aplikace bez podprocesů

```
xlC -q64 -o imqsput_64 imqsput.cpp -qchars=signed -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -limqs23ia -limqb23ia -lmqm
```

Aplikace s 64bitovým podprocesem

```
xlC_r -q64 -o imqsput_64_r imqsput.cpp -qchars=signed -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -limqs23ia_r -limqb23ia_r -lmqm_r
```

Sestavování programů C++ v systému HP-UX

Pomocí kompilátorů aC+ + nebo aCC sestavujte programy IBM MQ C++ na systému HP-UX .

V produktu HP-UX Itaniumpodporuje produkt IBM MQ pouze standardní běhový modul. Použijte kompilátor aCC .

- Soubor libimqi23bh.sl poskytuje třídy jazyka C++ produktu IBM MQ pro standardní běhový modul.
- Z důvodu kompatibility s dřívějšími verzemi je poskytnut symbolický odkaz z knihovny libimqi23ah.sl do souboru libimqi23bh.sl.

IA64 (IPF)

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Klient: IA64 (IPF)

32bitová aplikace bez podprocesů

```
aCC -Wl,+b,: +e -D_HPUX_SOURCE -o imqsputc_32 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -L/usr/lib/hpux32 -limqi23bh -lmqic
```

32bitovou aplikaci s podprocesy

```
aCC -Wl,+b,: +e -D_HPUX_SOURCE -o imqsputc_32_r imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -L/usr/lib/hpux32 -limqi23bh_r -lmqic_r -lpthread
```

64bitová aplikace bez podprocesů

```
aCC +DD64 +e -D_HPUX_SOURCE -o imqsputc_64 imqsput.cpp
-I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -limqi23bh
-lmqic
```

Aplikace s 64bitovým podprocesem

```
aCC +DD64 +e -D_HPUX_SOURCE -o imqsputc_64_r imqsput.cpp
-I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -limqi23bh_r
-lmqic_r
-lpthread
```

Server: IA64 (IPF)

32bitová aplikace bez podprocesů

```
aCC -Wl,+b,: +e -D_HPUX_SOURCE -o imqsput_32 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -L/usr/lib/hpux32 -limqi23bh -lmqm
```

32bitovou aplikaci s podprocesy

```
aCC -Wl,+b,: +e -D_HPUX_SOURCE -o imqsput_32_r imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -L/usr/lib/hpux32 -limqi23bh_r -lmqm_r -lpthread
```

64bitová aplikace bez podprocesů

```
aCC +DD64 +e -D_HPUX_SOURCE -o imqsput_64 imqsput.cpp
-I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -limqi23bh -lmqm
```

Aplikace s 64bitovým podprocesem

```
aCC +DD64 +e -D_HPUX_SOURCE -o imqsput_64_r imqsput.cpp
-I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -limqi23bh_r
-lmqm_r
-lpthread
```

Sestavování programů C++ v systému IBM i

Sestavte IBM MQ C++ programy na IBM i pomocí kompilátoru ILE C + +.

IBM ILE C++ for IBM i je nativní kompilátor pro programy C + +. Následující pokyny popisují způsob použití tohoto kompilátoru k vytváření aplikací jazyka C++ produktu IBM MQ pomocí *Hello World!* IBM MQ jako příklad.

1. Nainstalujte kompilátor ILE C + + pro kompilátor IBM i tak, jak je uvedeno v publikaci *Read Me first!* . příručka, která je přiložena k produktu.
2. Ujistěte se, že knihovna QCXXN je ve vašem seznamu knihoven.
3. Vytvořte ukázkový program HELLO WORLD:
   a. Vytvořte modul:

   ```
   CRTCPPMOD MODULE(MYLIB/IMQWRLD) +
   SRCSTMF('/QIBM/ProdData/mqm/samp/imqwrld.cpp') +
   INCDIR('/QIBM/ProdData/mqm/inc') DFTCHAR(*SIGNED) +
   TERASPACE(*YES)
   ```

   Zdroj pro ukázkové programy jazyka C++ lze najít v adresáři `/QIBM/ProdData/mqm/samp` a v souborech začlenění v produktu `/QIBM/ProdData/mqm/inc`.

   Případně je možné zdroj nalézt v knihovně `SRCFILE(QCPPSRC/LIB) SRCMBR(IMQWRLD)`.

   b. Svažte tuto kombinaci s programy služby dodanými s produktem IBM MQa vytvořte programový objekt:

   ```
   CRTPGM PGM(MYLIB/IMQWRLD) MODULE(MYLIB/IMQWRLD) +
   BNDSRVPGM(QMQM/IMQB23I4 QMQM/IMQS23I4)
   ```

   Chcete-li sestavit podprocesovou aplikaci, použijte reentrantní servisní programy:

   ```
   CRTPGM PGM(MYLIB/IMQWRLD) MODULE(MYLIB/IMQWRLD) +
   BNDSRVPGM(QMQM/IMQB23I4[_R] QMQM/IMQS23I4[_R])
   ```

   c. Spusťte vzorový program HELLO WORLD pomocí SYSTEM.DEFAULT.LOCAL.QUEUE:

```
CALL PGM(MYLIB/IMQWRLD)
```

Sestavování programů C++ v systému Linux

Sestavte programy IBM MQ C + + na Linux pomocí kompilátoru GNU g + +.

System p

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Klient:System p

32bitová aplikace bez podprocesů

```
g++ -m32 -o imqsputc_32 imqsput.cpp -fsigned-char -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqc23gl
-limqb23gl -lmqic
```

32bitovou aplikaci s podprocesy

```
g++ -m32 -o imqsputc_r32 imqsput.cpp -fsigned-char -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqc23gl_r
-limqb23gl_r -lmqic_r
```

64bitová aplikace bez podprocesů

```
g++ -m64 -o imqsputc_64 imqsput.cpp -fsigned-char -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -Wl,-rpath= MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqc23gl -limqb23gl -lmqic
```

Aplikace s 64bitovým podprocesem

```
g++ -m64 -o imqsputc_r64 imqsput.cpp -fsigned-char -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -Wl,-rpath= MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqc23gl_r -limqb23gl_r -lmqic_r
```

Server: System p

32bitová aplikace bez podprocesů

```
g++ -m32 -o imqsput_32 imqsput.cpp -fsigned-char -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqs23gl
-limqb23gl -lmqm
```

32bitovou aplikaci s podprocesy

```
g++ -m32 -o imqsput_r32 imqsput.cpp -fsigned-char -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqs23gl_r
-limqb23gl_r -lmqm_r
```

64bitová aplikace bez podprocesů

```
g++ -m64 -o imqsput_64 imqsput.cpp -fsigned-char -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -Wl,-rpath= MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqs23gl -limqb23gl -lmqm
```

Aplikace s 64bitovým podprocesem

```
g++ -m64 -o imqsput_r64 imqsput.cpp -fsigned-char -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -Wl,-rpath= MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqs23gl_r -limqb23gl_r -lmqm_r
```

IBM Z

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Klient:IBM Z

32bitová aplikace bez podprocesů

```
g++ -m31 -fsigned-char -o imqsputc_32 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqc23gl -limqb23gl -lmqic
```

32bitovou aplikaci s podprocesy

```
g++ -m31 -fsigned-char -o imqsputc_32_r imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqc23gl_r -limqb23gl_r -lmqic_r
-lpthread
```

64bitová aplikace bez podprocesů

```
g++ -m64 -fsigned-char -o imqsputc_64 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -Wl,-rpath= MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqc23gl -limqb23gl -lmqic
```

Aplikace s 64bitovým podprocesem

```
g++ -m64 -fsigned-char -o imqsputc_64_r imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -Wl,-rpath= MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqc23gl_r -limqb23gl_r -lmqic_r -lpthread
```

Server: IBM Z

32bitová aplikace bez podprocesů

```
g++ -m31 -fsigned-char -o imqsput_32 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqs23gl -limqb23gl -lmqm
```

32bitovou aplikaci s podprocesy

```
g++ -m31 -fsigned-char -o imqsput_32_r imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqs23gl_r -limqb23gl_r -lmqm_r -lpthread
```

64bitová aplikace bez podprocesů

```
g++ -m64 -fsigned-char -o imqsput_64 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -Wl,-rpath= MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqs23gl -limqb23gl -lmqm
```

Aplikace s 64bitovým podprocesem

```
g++ -m64 -fsigned-char -o imqsput_64_r imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -Wl,-rpath= MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqs23gl_r -limqb23gl_r -lmqm_r -lpthread
```

System x (32bitový)

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Klient: System x (32bitový)

32bitová aplikace bez podprocesů

```
g++ -m32 -fsigned-char -o imqsputc_32 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib -L
MQ_INSTALLATION_PATH/lib -Wl,
-rpath= MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -limqc23gl -limqb23gl -lmqic
```

32bitovou aplikaci s podprocesy

```
g++ -m32 -fsigned-char -o imqsputc_32_r imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib -L MQ_INSTALLATION_PATH/lib
-Wl,-rpath= MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -limqc23gl_r -limqb23gl_r
-lmqic_r -lpthread
```

64bitová aplikace bez podprocesů

```
g++ -m64 -fsigned-char -o imqsputc_64 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -Wl,-rpath= MQ_INSTALLATION_PATH/lib64 -L
MQ_INSTALLATION_PATH/lib64
-Wl,-rpath= MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64 -limqc23gl -limqb23gl
-lmqic
```

Aplikace s 64bitovým podprocesem

```
g++ -m64 -fsigned-char -o imqsputc_64_r imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -Wl,-rpath= MQ_INSTALLATION_PATH/lib64 -L
MQ_INSTALLATION_PATH/lib64
-Wl,-rpath= MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64 -limqc23gl_r -limqb23gl_r
-lmqic_r -lpthread
```

Server: System x (32bitový)

32bitová aplikace bez podprocesů

```
g++ -m32 -fsigned-char -o imqsput_32 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib -L MQ_INSTALLATION_PATH/lib
-Wl,-rpath= MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -limqs23gl -limqb23gl -lmqm
```

32bitovou aplikaci s podprocesy

```
g++ -m32 -fsigned-char -o imqsput_32_r imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib -L MQ_INSTALLATION_PATH/lib
-Wl,-rpath= MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -limqs23gl_r -limqb23gl_r
-lmqm_r -lpthread
```

64bitová aplikace bez podprocesů

```
g++ -m64 -fsigned-char -o imqsput_64 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -Wl,-rpath= MQ_INSTALLATION_PATH/lib64 -L
```

```
MQ_INSTALLATION_PATH/lib64
-Wl,-rpath= MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64 -limqs23gl -limqb23gl -lmqm
```

Aplikace s 64bitovým podprocesem

```
g++ -m64 -fsigned-char -o imqsput_64_r imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -Wl,-rpath= MQ_INSTALLATION_PATH/lib64 -L
MQ_INSTALLATION_PATH/lib64
-Wl,-rpath= MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64 -limqs23gl_r -limqb23gl_r
-lmqm_r -lpthread
```

Sestavování programů C++ v systému Solaris

Sestavte programy IBM MQ C + + na Solaris pomocí kompilátoru Sun ONE.

SPARC

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Klient: SPARC

32bitová aplikace

```
CC -xarch=v8plus -mt -o imqsputc_32 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -R MQ_INSTALLATION_PATH/lib -R/usr/lib/32 -limqc23as -limqb23as
-lmqic -lsocket -lnsl -ldl
```

64bitové aplikace

```
CC -xarch=v9 -mt -o imqsputc_64 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -R MQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -limqc23as
-limqb23as
-lmqic -lsocket -lnsl -ldl
```

Server: SPARC

32bitová aplikace

```
CC -xarch=v8plus -mt -o imqsput_32 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -R MQ_INSTALLATION_PATH/lib -R/usr/lib/32 -limqs23as -limqb23as
-lmqm -lsocket -lnsl -ldl
```

64bitové aplikace

```
CC -xarch=v9 -mt -o imqsput_64 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -R MQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -limqs23as
-limqb23as
-lmqm -lsocket -lnsl -ldl
```

x86-64

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Klient: x86-64

32bitová aplikace

```
CC -xarch=386 -mt -o imqsputc_32 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
```

```
-L MQ_INSTALLATION_PATH/lib -R MQ_INSTALLATION_PATH/lib -R/usr/lib/32 -limqc23as -limqb23as
-lmqic -lsocket -lnsl -ldl
```

64bitové aplikace

```
CC -xarch=amd64 -mt -o imqsputc_64 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -R MQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -limqc23as
-limqb23as
-lmqic -lsocket -lnsl -ldl
```

Server: x86-64

32bitová aplikace

```
CC -xarch=386 -mt -o imqsput_32 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -R MQ_INSTALLATION_PATH/lib -R/usr/lib/32 -limqs23as -limqb23as
-lmqm -lsocket -lnsl -ldl
```

64bitové aplikace

```
CC -xarch=amd64 -mt -o imqsput_64 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -R MQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -limqs23as
-limqb23as
-lmqm -lsocket -lnsl -ldl
```

Sestavování programů C++ v systému Windows

Sestavte programy IBM MQ C + + + na Windows pomocí kompilátoru jazyka C++ produktu Microsoft Visual Studio .

Upozornění: Knihovny dodané produktem IBM MQ jsou dynamické knihovny a ne statické knihovny. Produkt IBM MQ poskytuje informace známé jako "`import libraries`", které lze použít pouze během kompilačního času. Pro běhové prostředí musíte použít dynamické knihovny.

Z produktu IBM MQ 8.0.0 Fix Pack 4jsou produkty dodávány znovu distribuovatelným klientům, kteří obsahují knihovny potřebné pro spouštění aplikací produktu IBM MQ . Tyto knihovny mohou být zabaleny a redistribuovány s klientskými aplikacemi. Další informace najdete v tématu Redistribuovatelné klienty v systému Windows.

Soubory knihovny (.lib) a soubory dll pro použití s 32bitovými aplikacemi jsou nainstalovány v produktu *MQ_INSTALLATION_PATH*/Tools/Lib, soubory pro použití s 64bitovými aplikacemi jsou nainstalovány v produktu *MQ_INSTALLATION_PATH*/Tools/Lib64. *MQ_INSTALLATION_PATH* představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Klient

```
cl -MD imqsput.cpp /Feimqsputc.exe imqb23vn.lib imqc23vn.lib
```

Server

```
cl -MD imqsput.cpp /Feimqsput.exe imqb23vn.lib imqs23vn.lib
```

Knihovny klienta C++ sestavené pomocí kompilátoru Microsoft Visual Studio 2015

V 9.0.1

V 9.0.1 Z produktu IBM MQ 9.0.1poskytuje produkt knihovny klienta C + +, které jsou sestaveny s použitím kompilátoru jazyka C++ produktu Microsoft Visual Studio 2015 . Aplikace, které jsou sestaveny pomocí vydání produktu IBM MQ 9.0.1 nebo pozdějších, mohou používat tyto knihovny. Tyto knihovny jsou

k dispozici spolu s existujícími knihovnami produktu IBM MQ 9.0.1 C + +, které jsou sestaveny pomocí kompilátoru jazyka C++ produktu Microsoft Visual Studio 2012 .

Both 32-bit and 64-bit versions of the IBM MQ C++ libraries are provided. I když jsou 32bitové knihovny instalovány ve složce `bin\vs2015` , 64bitové knihovny se instalují do složek produktu `bin64\vs2015` .

Produkt IBM MQ je standardně nakonfigurován pro použití knihoven Microsoft Visual Studio 2012 . Chcete-li použít knihovny produktu Microsoft Visual Studio 2015 , je třeba nastavit proměnnou prostředí `MQ_PREFIX_VS_LIBRARIES` pomocí příkazu **setmqenv**nebo **setmqinst** .

Můžete sestavovat vlastní aplikace systému zpráv s produktem Microsoft Visual Studio 2017 a propojit je s dodanými knihovnami produktu IBM MQ C/C++ Microsoft Visual Studio 2015 .

Použití produktu IBM MQ s kompilátorem jazyka C++ produktu Microsoft Visual Studio 2015 .

V produktu Microsoft Visual Studio 2015sloučené moduly produktu Microsoft nainstalované s produktem IBM MQ již neobsahují celý běhový kód jazyka C.

Chcete-li použít kompilátor jazyka C++ produktu Microsoft Visual Studio 2015 , musíte instalovat aktualizaci produktu Microsoft Knowledge Base, KB3118401, z webové stránky Windows 10 Universal C Runtime , pokud používáte verzi produktu Windows před verzí Windows 10.

Tato stránka obsahuje systémové požadavky a pokyny k instalaci.

Pokud neinstalujete KB3118401, programy C + + založené na produktu Microsoft Visual Studio 2015 (jak IBM , tak pro váš vlastní podnik) selžou při spuštění, obvykle se objeví následující zpráva:

```
The program can't start because api-ms-win-crt-runtime-|1-1-0.dll
is missing from your computer. Try reinstalling the program to
fix this problem.
```

Použití odlišně pojmenovaných knihoven IBM MQ C + +

V produktu IBM MQ 8.0.0 Fix Pack 4produkt poskytuje některé další knihovny klienta C + +, které jsou pojmenovány odlišně. Tyto knihovny jsou sestaveny pomocí kompilátoru jazyka C++ produktu Microsoft Visual Studio 2012 . K dispozici jsou také knihovny produktu Microsoft Visual Studio 2015 . Tyto knihovny jsou k dispozici spolu s existujícími knihovnami jazyka C + +, které jsou také sestaveny s použitím kompilátoru Microsoft Visual Studio 2012 nebo Microsoft Visual Studio 2015 C + +. Vzhledem k tomu, že tyto další knihovny jazyka C++ produktu IBM MQ mají různé názvy, můžete spustit aplikace produktu IBM MQ C + +, sestavené pomocí produktu IBM MQ C++ a kompilovány s produktem Microsoft Visual Studio 2012 a staršími verzemi produktu na stejném počítači.

Další knihovny produktu Microsoft Visual Studio 2012 jsou pojmenovány následujícím způsobem:

- `imqb23vnvs2012.dll`
- `imqc23vnvs2012.dll`
- `imqs23vnvs2012.dll`
- `imqx23vnvs2012.dll`

Další knihovny produktu Microsoft Visual Studio 2015 jsou pojmenovány následujícím způsobem:

- `imqb23vnvs2015.dll`
- `imqc23vnvs2015.dll`
- `imqs23vnvs2015.dll`
- `imqx23vnvs2015.dll`

Je poskytováno 32bitové i 64bitové verze těchto knihoven. 32-bitové knihovny jsou instalovány ve složce `bin` a 64bitové knihovny jsou instalovány ve složce `bin64` . Odpovídající knihovny importu jsou nainstalovány v adresářích `Tools\lib` a `Tools\lib64` .

Pokud vaše aplikace používá soubory `imq*vs2012.lib` , musíte ji zkompilovat pomocí kompilátoru Microsoft Visual Studio 2012 . Chcete-li spustit aplikace IBM MQ C + +, které jsou kompilovány s produktem Microsoft Visual Studio 2012 a aplikacemi, které jsou kompilovány s dřívější verzí produktu na stejném počítači, musí být proměnná prostředí PATH uvedena jako předpona, jak je uvedeno v následujících příkladech:

- Pro 32 bitové aplikace:

```
SET PATH=installation folder\bin\vs2008;%PATH%
```

- Pro 64bitové aplikace:

```
SET PATH=installation folder\bin64\vs2008;%PATH%
```

Související informace

Windows: změny pro IBM MQ 8.0

Sestavování programů jazyka C++ v produktu z/OS Batch, Dávka RRS a CICS

Sestavte IBM MQ C++ programy na z/OS pro dávku, dávku RRS nebo prostředí CICS a spusťte ukázkové programy.

Můžete napsat programy C + + pro tři z prostředí, která produkt IBM MQ for z/OS podporuje:

- Dávka
- Dávka RRS
- CICS

Kompilace, předběžné propojení a propojení

Vytvoření aplikace z/OS kompilací, předlinkování a linkování-editování zdrojového kódu C++.

Produkt IBM MQ C++ for z/OS je implementován jako z/OS DLL pro jazyk IBM C++ pro jazyk z/OS . Použijete-li knihovny DLL, zřetězíte dodané definiční sady s výstupem kompilátoru v době před propojením. This allows the linker to check your calls to the IBM MQ C++ member functions.

Poznámka: Pro každý ze tří prostředí jsou k dispozici tři sady bočnic.

Chcete-li sestavit aplikaci IBM MQ for z/OS C + +, vytvořte a spusťte JCL. Postupujte takto:

1. Pokud se vaše aplikace spouští v produktu CICS, použijte k převodu CICS příkazů ve vašem programu proceduru dodávanou s produktem CICS.

   Kromě toho pro aplikace CICS je třeba:

   a. Přidejte knihovnu SCSQLOAD do zřetězení DFHRPL.

   b. Definujte skupinu CSQCAT1 CEDA za použití členu IMQ4B100 v knihovně SCSQPROC.

   c. Nainstalujte produkt CSQCAT1.
2. Zkompilujte program a vytvořte kód objektu. Kód JCL pro vaši kompilaci musí obsahovat příkazy, které zpřístupní soubory definic dat produktu kompilátoru. Definice dat jsou dodávány v následujících knihovnách produktu IBM MQ for z/OS :

   - **thlqual**.SCSQC370
   - **thlqual**.SCSQHPS

   Ujistěte se, že jste zadali volbu kompilátoru `/cxx` .

   Poznámka: Název **thlqual** je kvalifikátor vysoké úrovně instalační knihovny produktu IBM MQ v systému z/OS.
3. Předpropojte objektový kód vytvořený v kroku “2” na stránce 512, včetně následujících definičních ploch, které jsou dodávány v **thlqual**.SCSQDEFS:

a. imqs23dm a imqb23dm pro dávku

b. Soubor imqs23dr a imqb23dr pro dávku RRS

c. imqs23dc a imqb23dc pro CICS

Jedná se o odpovídající knihovny DLL.

a. imqs23im a imqb23im pro dávku

b. imqs23ir a imqb23ir pro dávku RRS

c. imqs23ic a imqb23ic pro CICS

4. Odkaz-upravte kód objektu vytvořený v kroku “3” na stránce 512, vytvořte zaváděcí modul a uložte jej do zaváděcí knihovny aplikací.

Chcete-li spustit dávkové programy nebo dávkové programy RRS, zahrňte do zřetězení datové sady STEPLIB nebo JOBLIB knihovny **thlqual**.SCSQAUTH a **thlqual**.SCSQLOAD.

To run a CICS program, first get your system administrator to define it to CICS as an IBM MQ program and transaction. Poté ji můžete spustit obvyklým způsobem.

Spustit ukázkové programy

Tyto programy jsou popsány v příručce “Ukázkové programy C++” na stránce 488.

Ukázkové aplikace jsou dodávány pouze ve zdrojovém tvaru. Soubory jsou:

Tabulka 74. Ukázkové programové soubory produktu z/OS

Ukázka	Zdrojový program (v knihovně thlqual.SCSQCPPS)	JCL (v knihovně thlqual.SCSQPROC)
Hello World	imqwrld	imqwrldr
SUT	imqsput	imqsputr
SGET	imqsget	imqsgetr

Chcete-li spustit ukázky, zkompilovat a propojit je jako s jakýmkoli programem C + + (viz “Sestavování programů jazyka C++ v produktu z/OS Batch, Dávka RRS a CICS” na stránce 512). K vytvoření a spuštění dávkové úlohy použijte dodaný JCL. JCL musíte nejprve upravit, a to pomocí následujícího komentáře, který je součástí tohoto souboru.

Sestavování programů jazyka C++ v produktu z/OS UNIX System Services

Sestavte programy IBM MQ C + + v systému z/OS for Unix System Services.

Chcete-li sestavit aplikaci pod shellem systémových služeb produktu UNIX , musíte poskytnout přístup kompilátoru k souborům produktu IBM MQ (umístěným v adresáři `thlqual.SCSQC370` a `hlqual.SCSQHPPS`) a odkazovat na dvě z jednotek knihovny DLL (které se nacházejí v produktu `thlqual.SCSQDEFS`). V běhovém prostředí potřebuje aplikace přístup k datovým sadám produktu IBM MQ `thlqual.SCSQLOAD`, `thlqual.SCSQAUTH`a jednomu z datových sad specifických pro jazyk, jako je například `thlqual.SCSQANLE`[6].

Kompilace

1. Zkopírujte vzorek do systému souborů HFS pomocí příkazu TSO **oput** nebo použijte protokol FTP. Zbytek tohoto příkladu předpokládá, že jste zkopírovali ukázku do adresáře s názvem `/u/fred/samplea` pojmenovali jej `imqwrld.cpp`.

[6] Chcete-li spustit svou systémovou službu UNIX v libovolném ze tří prostředí, můžete propojit s některou z bočnic uvedených v produktu " Předběžný odkaz na kód objektu . “Sestavování programů jazyka C++ v produktu z/OS Batch, Dávka RRS a CICS” na stránce 512

2. Přihlaste se do shellu produktu UNIX System Services a přejděte do adresáře, do kterého jste umístili ukázku.
3. Nastavte kompilátor jazyka C++ tak, aby mohl jako vstup přijmout soubory DLL sidedeck a .cpp:

```
/u/fred/sample:> export _CXX_EXTRA_ARGS=1
/u/fred/sample:> export _CXX_CXXSUFFIX="cpp"
```

4. Kompilujte a propojte ukázkový program. Následující příkaz propojí program s dávkovými palubami; místo toho lze použít dávková postranní plochy RRS. Znak \ se používá k rozdělení příkazu na více než jeden řádek. Nezadávejte tento znak; zadejte příkaz jako jeden řádek:

```
/u/fred/sample:> c++ -o imqwrld -I "//'thlqual.SCSQC370'" \
-I "//'thlqual.SCSQHPPS'" imqwrld.cpp \
"//'thlqual.SCSQDEFS(IMQS23DM)'" "//'thlqual.SCSQDEFS(IMQB23DM)'"
```

Další informace o příkazu TSO **oput** naleznete v příručce *z/OS UNIX System Services Command Reference*.

Pomocí obslužného programu make můžete také zjednodušit sestavování programů v jazyce C + +. Zde je ukázkový soubor Makefile pro sestavení ukázkového programu HELLO WORLD C++. Odděluje fáze kompilace a linkování. Nastavte prostředí jako v kroku “3” na stránce 514 před spuštěním make.

```
flags = -I "//'thlqual.SCSQC370'" -I "//'thlqual.SCSQHPPS'"
decks = "//'thlqual.SCSQDEFS(IMQS23DM)'" "//'thlqual.SCSQDEFS(IMQB23DM)'"

imqwrld: imqwrld.o
   c++ -o imqwrld imqwrld.o $(decks)

imqwrld.o: imqwrld.cpp
   c++ -c -o imqwrld $(flags) imqwrld.cpp
```

Další informace o použití make naleznete v příručce *Programovací nástroje systémových služeb produktu z/OS UNIX* .

Spuštěno

1. Přihlaste se do shellu produktu UNIX System Services a přejděte do adresáře, do kterého jste sestavení vytvořili.
2. Nastavte proměnnou prostředí STEPLIB tak, aby obsahovala datové sady produktu IBM MQ :

```
/u/fred/sample:> export STEPLIB=$STEPLIB:thlqual.SCSQLOAD
/u/fred/sample:> export STEPLIB=$STEPLIB:thlqual.SCSQAUTH
/u/fred/sample:> export STEPLIB=$STEPLIB:thlqual.SCSQANLE
```

3. Spusťte ukázku:

```
/u/fred/sample:> ./imqwrld
```

Vývoj aplikací produktu .NET

IBM MQ classes for .NET umožňuje programu zapsaným v programovacím rámci .NET pro připojení k serveru IBM MQ jako IBM MQ MQI client nebo k přímému připojení k serveru IBM MQ .

Máte-li aplikace, které používají rámec Microsoft .NET Framework a chcete využívat výhody zařízení produktu IBM MQ, je třeba použít produkt IBM MQ classes for .NET.

Rozhraní objektově orientované IBM MQ .NET se liší od rozhraní MQI v tom, že používá metody objektů spíše než při použití příkazových slov MQI.

Procedurální programovací rozhraní produktu IBM MQ je založeno na příkazových slovech, jako jsou např. v následujícím seznamu:

```
MQCONN, MQDISC, MQOPEN, MQCLOSE,
MQINQ, MQSET, MQGET, MQPUT, MQSUB
```

Tato příkazová slova slouží jako argument pro obsluhu objektu IBM MQ , na kterém mají pracovat. Protože .NET je objektově orientovaná, toto zaokrouhlení se změní na programovací rozhraní .NET . Váš program se skládá ze sady objektů IBM MQ , které se při volání metod na těchto objektech chovají. Programy můžete psát v libovolném jazyce, který je podporován produktem .NET.

Když použijete procedurální rozhraní, odpojíte se od správce front pomocí volání MQDISC (*Hconn*, CompCode, Reason), kde *Hconn* je manipulátor pro správce front.

V rozhraní produktu .NET je správce front reprezentován objektem třídy MQQueueManager. Odpojení od správce front voláte voláním metody Disconnect () na dané třídě.

```
// declare an object of type queue manager
MQQueueManager queueManager=new MQQueueManager();
...
// do something...
...
// disconnect from the queue manager
queueManager.Disconnect();
```

IBM MQ classes for .NET je sada tříd, které umožňují aplikacím produktu .NET interakci s produktem IBM MQ. Představují různé komponenty produktu IBM MQ , které vaše aplikace používá, jako jsou správci front, fronty, kanály a zprávy. Podrobné informace o těchto třídách najdete v tématu Třídy a rozhraní produktu IBM MQ .NET.

Než budete moci zkompilovat všechny aplikace, které napíšete, musíte mít nainstalován produkt .NET Framework. Pokyny k instalaci produktu IBM MQ classes for .NET a rámce .NET Framework viz “instalaceIBM MQ classes for .NET” na stránce 516.

Související pojmy

“Volby pro připojování ke správci front” na stránce 516
K dispozici jsou tři režimy připojení IBM MQ classes for .NET ke správci front. Zvažte, který typ připojení nejlépe vyhovuje vašim požadavkům.

“Zápis a implementace programů produktu IBM MQ .NET” na stránce 529
Chcete-li použít produkt IBM MQ classes for .NET pro přístup k frontám IBM MQ , naprograte programy v libovolném jazyce, který je podporován produktem .NET a který obsahuje zprávy o vkládání zpráv do front zpráv a získávání zpráv z front produktu IBM MQ .

“Vývoj aplikací Microsoft Windows Communication Foundation (WCF) s IBM MQ” na stránce 1217
Vlastní kanál Microsoft Windows Communication Foundation (WCF) pro produkt IBM MQ odesílá a přijímá zprávy mezi klienty WCF a službami.

“Vyvíjení aplikací pro IBM MQ” na stránce 5
Můžete vyvíjet aplikace k odesílání a přijímání zpráv a ke správě správců front a souvisejících prostředků. IBM MQ podporuje aplikace napsané v mnoha různých jazycích a rámcích.

Související informace

Technický přehled
Odstraňování problémů s produktem IBM MQ.NET

Začínáme s produktem IBM MQ classes for .NET

IBM MQ classes for .NET umožňuje programu zapsaným v programovacím rámci .NET pro připojení k serveru IBM MQ jako IBM MQ MQI client nebo k přímému připojení k serveru IBM MQ .

Volby pro připojování ke správci front

K dispozici jsou tři režimy připojení IBM MQ classes for .NET ke správci front. Zvažte, který typ připojení nejlépe vyhovuje vašim požadavkům.

Připojení vazeb klienta

Chcete-li použít produkt IBM MQ classes for .NET jako IBM MQ MQI client, můžete jej instalovat s prostorem IBM MQ MQI clientbuď na počítači serveru IBM MQ , nebo na samostatném počítači. Připojení vazeb klienta může využívat transakce XA nebo non-XA.

Připojení vazeb serveru

Při použití v režimu vazeb serveru IBM MQ classes for .NET používá rozhraní API správce front, a nikoli komunikaci prostřednictvím sítě. To poskytuje lepší výkon pro aplikace produktu IBM MQ než použití síťových připojení.

Chcete-li používat připojení vazeb, je třeba nainstalovat produkt IBM MQ classes for .NET na server IBM MQ .

Připojení spravovaného klienta

Připojení vytvořené v tomto režimu se připojuje jako klient IBM MQ k serveru IBM MQ spuštěnému buď na lokálním, nebo na vzdáleném počítači.

IBM MQ classes for .NET připojení v tomto režimu zůstává ve spravovaném kódu .NET a nevyžaduje žádné volání nativních služeb. Další informace o spravovaném kódu naleznete v dokumentaci produktu Microsoft .

Pro použití spravovaného klienta existuje několik omezení. Další informace o těchto viz “Připojení spravovaného klienta” na stránce 529.

instalaceIBM MQ classes for .NET

IBM MQ classes for .NET, včetně ukázek, je nainstalováno s produktem IBM MQ. Existuje předpoklad produktu Microsoft.NET Framework.

Jako součást standardní instalace produktu IBM MQ v rámci funkce *Java a .NET Messaging and Web Services* je standardně nainstalována nejnovější verze produktu IBM MQ classes for .NET . Pokyny k instalaci naleznete v tématu Instalace serveru IBM MQ v systému Windows nebo Instalace klienta IBM MQ v systémech Windows.

Pokud jste v prostředí s více instalačními prostředí dříve nainstalovali produkt IBM MQ classes for .NET jako balík podpory, nemůžete instalovat produkt IBM MQ , pokud nejprve neodinstalujete balík podpory. Funkce produktu IBM MQ classes for .NET , která je nainstalována s produktem IBM MQ , obsahuje stejnou funkčnost jako balík podpory.

K dispozici jsou také ukázkové aplikace, včetně zdrojových souborů, viz “Ukázkové aplikace” na stránce 517.

Chcete-li spustit produkt IBM MQ classes for .NET na 32bitových nebo 64bitových platformách, musíte mít nainstalován produkt Microsoft.NET Framework V3.5 nebo novější.

Poznámka: Není-li před instalací produktu IBM MQ 8.0nainstalován produkt Microsoft.NET Framework v3.5 nebo vyšší, bude instalace produktu IBM MQ bez chyb pokračovat, ale IBM MQ classes for .NET nebude k dispozici. Je-li produkt .NET Framework nainstalován po instalaci produktu IBM MQ 8.0, musí být sestavy IBM MQ.NET registrovány spuštěním skriptu *WMQInstallDir*`\bin\amqiRegisterdotNet.cmd` , kde *WMQInstallDir* je adresář, kde je nainstalován produkt IBM MQ 8.0 . Tento skript nainstaluje vyžadované sestavení v mezipaměti GAC (Global Assembly Cache). Sada souborů `amqi*.log` zaznamenávající provedené akce se vytvoří v adresáři %TEMP% .

Informace o použití vlastního kanálu produktu IBM MQ pro prostředek Microsoft WCF s produktem .NET 3 naleznete v tématu “Vývoj aplikací Microsoft Windows Communication Foundation (WCF) s IBM MQ” na stránce 1217 .

Ukázkové aplikace

Chcete-li spustit vlastní aplikace produktu .NET , použijte pokyny pro programy ověření nahrazující název aplikace v místě ukázkových aplikací.

Dodává se pět ukázkových aplikací:

- Aplikace pro vkládání zpráv
- Aplikace pro získání zprávy
- Aplikace 'Ahoj světe'
- Aplikace typu publikování/odběr.
- Aplikace používající vlastnosti zprávy

Všechny tyto ukázkové aplikace jsou dodávány v jazyce C# a některé jsou také dodávány v jazycích C++ a Visual Basic. Aplikace můžete psát v libovolném jazyce, který je podporován produktem .NET.

"Vložit zprávu" program SPUT (nmqsput.cs, mmqsput.cpp, vmqsput.vb)
Tento program ukazuje, jak vložit zprávu do pojmenované fronty. Program má tři parametry:

- Název fronty (povinné), například SYSTEM.DEFAULT.LOCAL.QUEUE
- Název správce front (volitelný)
- Definice kanálu (volitelné), například SYSTEM.DEF.SVRCONN/TCP/hostname(1414)

Není-li zadán žádný název správce front, správce front se standardně nastaví na výchozího lokálního správce front. Je-li kanál definován, má stejný formát jako proměnná prostředí MQSERVER.

"Get message" program SGET (nmqsget.cs, mmqsget.cpp, vmqsget.vb)
Tento program ukazuje, jak získat zprávu z pojmenované fronty. Program má tři parametry:

- Název fronty (povinné), například SYSTEM.DEFAULT.LOCAL.QUEUE
- Název správce front (volitelný)
- Definice kanálu (volitelné), například SYSTEM.DEF.SVRCONN/TCP/hostname(1414)

Není-li zadán žádný název správce front, správce front se standardně nastaví na výchozího lokálního správce front. Je-li kanál definován, má stejný formát jako proměnná prostředí MQSERVER.

Program "Ahoj světe" (nmqwrld.cs, mmqwrld.cpp, vmqwrld.vb)
Tento program ukazuje, jak vložit a získat zprávu. Program má tři parametry:

- Název fronty (volitelné), například SYSTEM.DEFAULT.LOCAL.QUEUE nebo SYSTEM.DEFAULT.MODEL.QUEUE
- Název správce front (volitelný)
- Definice kanálu (volitelná), například SYSTEM.DEF.SVRCONN/TCP/hostname(1414)

Není-li zadán žádný název fronty, bude použit výchozí název SYSTEM.DEFAULT.LOCAL.QUEUE. Není-li zadán žádný název správce front, správce front se standardně nastaví na výchozího lokálního správce front.

Program "Publish/subscribe" (MQPubSubSample.cs)
Tento program ukazuje, jak používat publikaci IBM MQ publish/subscribe. Dodává se pouze v C#. Program má dva parametry:

- Název správce front (volitelný)
- Definice kanálu (volitelná).

Program vlastností "Message properties" (MQMessagePropertiesSample.cs)
Tento program ukazuje, jak používat vlastnosti zpráv. Dodává se pouze v C#. Program má dva parametry:

- Název správce front (volitelný)
- Definice kanálu (volitelná).

Vaši instalaci můžete ověřit kompilací a spuštěním těchto aplikací.

Ukázkové aplikace jsou instalovány do následujících umístění podle jazyka, v němž jsou napsány. *MQ_INSTALLATION_PATH* představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

C#

MQ_INSTALLATION_PATH\Tools\dotnet\samples\cs\nmqswrld.cs

MQ_INSTALLATION_PATH\Tools\dotnet\samples\cs\nmqsput.cs

MQ_INSTALLATION_PATH\Tools\dotnet\samples\cs\nmqsget.cs

MQ_INSTALLATION_PATH\Tools\dotnet\samples\cs\MQPubSubSample.cs

MQ_INSTALLATION_PATH\Tools\dotnet\samples\cs\MQMessagePropertiesSample.cs

Managed C++

MQ_INSTALLATION_PATH\Tools\dotnet\samples\mcp\mmqswrld.cpp

MQ_INSTALLATION_PATH\Tools\dotnet\samples\mcp\mmqsput.cpp

MQ_INSTALLATION_PATH\Tools\dotnet\samples\mcp\mmqsget.cpp

Visual Basic

MQ_INSTALLATION_PATH\Tools\dotnet\samples\vb\vmqswrld.vb

MQ_INSTALLATION_PATH\Tools\dotnet\samples\vb\vmqsput.vb

MQ_INSTALLATION_PATH\Tools\dotnet\samples\vb\vmqsget.vb

MQ_INSTALLATION_PATH\Tools\dotnet\samples\vb\xmqswrld.vb

MQ_INSTALLATION_PATH\Tools\dotnet\samples\vb\xmqsput.vb

MQ_INSTALLATION_PATH\Tools\dotnet\samples\vb\xmqsget.vb

Chcete-li sestavit ukázkové aplikace, byl pro každý jazyk dodán dávkový soubor.

C#

MQ_INSTALLATION_PATH\Tools\dotnet\samples\cs\bldcssamp.bat

Soubor bldcssamp.bat obsahuje řádek pro každý vzorek, což je vše, co je nezbytné k sestavení tohoto ukázkového programu:

```
csc /t:exe /r:System.dll /r:amqmdnet.dll /lib: MQ_INSTALLATION_PATH\bin
/out:nmqwrld.exe nmqwrld.cs
```

Managed C++

MQ_INSTALLATION_PATH\Tools\dotnet\samples\mcp\bldmcpsamp.bat

Soubor bldmcpsamp.bat obsahuje řádek pro každý vzorek, což je vše, co je nezbytné k sestavení tohoto ukázkového programu:

```
cl /clr:oldsyntax MQ_INSTALLATION_PATH\bin mmqwrld.cpp
```

Chcete-li tyto aplikace zkompilovat v produktu Microsoft Visual Studio 2003/.NET SDKv1.1, nahraďte kompilační příkaz:

```
cl /clr:oldsyntax MQ_INSTALLATION_PATH\bin mmqwrld.cpp
```

s

```
cl /clr MQ_INSTALLATION_PATH\bin mmqwrld.cpp
```

Visual Basic

MQ_INSTALLATION_PATH\Tools\dotnet\samples\vb\bldvbsamp.bat

Soubor bldvbsamp.bat obsahuje řádek pro každý vzorek, což je vše, co je nezbytné k sestavení tohoto ukázkového programu:

```
vbc /r:System.dll /r: MQ_INSTALLATION_PATH\bin\amqmdnet.dll /out:vmqwrld.exe vmqwrld.vb
```

Konfigurace správce front tak, aby přijímal klientská připojení TCP/IP

Nakonfigurujte správce front tak, aby přijímal příchozí požadavky na připojení od klientů.

Informace o této úloze

Tato úloha vysvětluje základní kroky konfigurace správce front tak, aby přijímal klientská připojení TCP/IP. V případě provozního systému je třeba při konfigurování správců front zvážit také důsledky zabezpečení.

Postup

1. Definujte kanál připojení serveru:
   a. Spusťte správce front.
   b. Definujte ukázkový kanál s názvem NET.CHANNEL:

   ```
   DEF CHL('NET.CHANNEL') CHLTYPE(SVRCONN) TRPTYPE(TCP) MCAUSER(' ') +
   DESCR('Sample channel for IBM MQ classes for .NET')
   ```

   Důležité: Tento vzorek je určen pouze pro použití v prostředí sandboxů, protože neobsahuje žádné úvahy o důsledcích zabezpečení. V případě produkčního systému zvažte použití zabezpečení TLS nebo procedury zabezpečení. Další informace naleznete v dokumentu Zabezpečení IBM MQ .
2. Spustit modul listener:

   ```
   runmqlsr -t tcp [-m qmnqme ] [-p portnum ]
   ```

   Poznámka: Hranaté závorky označují volitelné parametry; *qmname* není vyžadováno pro výchozího správce front a číslo portu *číslo_portu* není povinné, pokud používáte výchozí hodnotu (1414).

Distribuované transakce v produktu .NET

Distribuované transakce nebo globální transakce umožňují klientským aplikacím zahrnout do jedné transakce několik různých zdrojů dat na dvou nebo více síťových systémech.

V distribuovaných transakcích správce transakcí koordinuje a spravuje transakci mezi dvěma či více správci prostředků.

Transakce mohou být buď jednofázové, nebo dvoufázové procesy vázaného zpracování. Jednofázové potvrzení je proces, jehož jediným správcem prostředků je zapojen do procesu transakce a dvoufázového potvrzování, je-li v transakci více než jeden správce prostředků, který se podílí na transakci. Ve dvoufázovém procesu potvrzování odešle správce transakcí připravenou výzvu ke kontrole, zda jsou všichni správci prostředků připraveni k potvrzení. Když přijme potvrzení od všech správců prostředků, vydá se výzva k potvrzení. V opačném případě dojde k odvolání transakce při celé transakci. Další podrobnosti viz Správa transakcí a podpora . Správci prostředků by měli informovat správce transakcí o jejich účasti v transakci. Když správce prostředků informuje správce transakcí o své účasti, získá správce prostředků zpětné volání od správce transakcí, když transakce potvrdí nebo vrátí zpět.

Třídy IBM MQ .NET již podporují distribuované transakce v nespravovaných připojeních a v režimu vazeb serveru. V těchto režimech produkt IBM MQ .NET deleguje všechna volání klienta rozšířených transakcí jazyka C, který spravuje zpracování transakce jménem produktu .NET.

Třídy IBM MQ.NET nyní podporují distribuované transakce ve spravovaném režimu, kde produkt IBM MQ .NET Classes používá obor názvů System.Transactions pro podporu distribuovaných transakcí. Infrastruktura System.Transactions usnadňuje a zefektivňuje transakční programování tím, že podporuje transakce zahájené ve všech správcích prostředků včetně produktu IBM MQ. Aplikace IBM MQ .NET může vkládat a získávat zprávy pomocí implicitního programování transakcí .NET nebo explicitního modelu programování transakcí. V implicitních transakcích jsou hranice transakce vytvářeny aplikačním programem, který rozhoduje o tom, kdy se má potvrdit, odvolat (pro explicitní transakce) nebo dokončit transakci. V explicitních transakcích musíte výslovně uvést, zda chcete potvrdit, odvolat a dokončit transakci.

IBM MQ.NET uses Microsoft distributed transaction coordinator (MS DTC) as the transaction manager, which coordinates and manages the transaction between multiple resource managers. IBM MQ se používá jako správce prostředků. Všimněte si, že nemůžete použít TLS s transakcemi XA. Musíte použít tabulky CCDT. Další informace naleznete v tématu Použití rozšířeného transakčního klienta s kanály TLS.

IBM MQ.NET následuje model X/Open Distributed Transaction Processing (DTP). Model distribuovaného zpracování transakcí X/Open je distribuovaným modelem zpracování transakcí navrženým skupinou Open Group, konsorciem dodavatele. Tento model je standardem většiny komerčních dodavatelů v transakčním zpracování transakcí a v doménách databáze. Většina komerčních produktů pro správu transakcí podporuje model X/DTP.

Způsoby transakce

- “Distribuované transakce ve spravovaném režimu” na stránce 521
- Distribuované transakce pro nespravovaný režim

Koordinace transakcí v různých scénářích

- Připojení se může podílet na několika transakcích, ale v libovolném časovém okamžiku je aktivní pouze jedna transakce.
- Během transakce je to MQQueueManager.Volání odpojení bylo uznáno. V tomto případě je transakce požádána o odvolání transakce.
- V průběhu transakce je volání MQQueue.Close nebo MQTopic.Close uznáno. V tomto případě je transakce požádána o odvolání transakce.
- Hranice transakce jsou vytvářeny aplikačním programem, který rozhoduje o tom, kdy se má potvrdit, odvolat (pro explicitní transakce) nebo dokončit transakci (pro implicitní transakce).
- Pokud se aplikace klienta během transakce přeruší s neočekávanou chybou před zadáním volání příkazu Put nebo Get ve volání fronty nebo tématu, transakce je odvolána a dojde k vyvolání výjimky MQException.
- Je-li kód příčiny MQCC_FAILED vrácen během volání funkce Put nebo Get v rámci volání fronty nebo tématu, dojde k výjimce MQException s kódem příčiny a transakce se bude přehrávány. Pokud správce transakcí již vydal připravovací volání, produkt IBM MQ .NET vrátí požadavek na přípravu tím, že provede vynucené odvolání transakce. Poté správce transakcí DTC vyvolá odvolání změn v aktuální práci se všemi správci prostředků v aktuálních okolních transakcích.
- Pokud během transakce zahrnující více správců prostředků způsobí, že některý z důvodů ochrany životního prostředí způsobí, že se operace Put nebo Get budou pozastaveny na dobu neurčitou, bude správce transakcí čekat až do stanoveného času. Po vypršení časového limitu dojde k odvolání veškeré aktuální práce se všemi správci prostředků v aktuálních okolních transakcích. Pokud toto čekání na dobu neurčitou nastane během fáze přípravy, správce transakcí může vypršet nebo může vyvolat nejisté volání na prostředku v takovém případě, že transakce je odvolána.

- Aplikace používající transakce musí vložit nebo získat zprávy pod SYNC_POINT. Je-li v rámci transakčního kontextu, který není pod názvem SYNC_POINT, vydána zpráva s voláním funkce Put nebo Get, volání selže s kódem příčiny MQRC_UNIT_OF_WORK_NOT_STARTED.

Rozdíly v chování mezi podporou spravovaných a nespravovaných klientských transakcí pomocí oboru názvů System.Transactions produktu Microsoft.NET

Vnořené transakce mají objekt TransactionScope uvnitř jiného TransactionScope

- Plně spravovaný klient produktu IBM MQ .NET podporuje vnořené TransactionScope
- Nespravovaný klient IBM MQ .NET nepodporuje vnořený objekt TransactionScope

Závislé transakce z System.Transactions

- Plně spravovaný klient IBM MQ .NET podporuje poskytovanou službu transakcí poskytovanou prostřednictvím System.Transactions.
- Nespravovaný klient IBM MQ .NET nepodporuje prostředek závislých transakcí poskytovaný produktem System.Transactions.

Ukázky produktu

Nové ukázky produktů `SimpleXAPuta SimpleXAGet` jsou dostupné v části `WebSphere MQ\tools\dotnet\samples\cs\base`. Ukázky jsou C# aplikace, které demonstrují pomocí MQPUT a MQGET pod Distribuovanými transakcemi pomocí oboru názvů SystemTransactions . Další informace o těchto ukázkách naleznete v tématu “Vytváření jednoduchých vložení a získání zpráv v rámci TransactionScope” na stránce 524 .

Distribuované transakce ve spravovaném režimu

Třídy IBM MQ .NET používají obor názvů System.Transactions pro podporu distribuovaných transakcí ve spravovaném režimu. Ve spravovaném režimu koordinuje MS DTC souřadnice a spravuje distribuované transakce na všech serverech uvedených v transakci.

Třídy IBM MQ .NET poskytují explicitní programovací model založený na třídě System.Transactions.Transaction a implicitním programovacím modelu za použití System.Transactions.TransactionScope, třída, kde jsou transakce automaticky spravovány infrastrukturou.

Implicitní transakce

Následující část kódu popisuje způsob, jakým aplikace IBM MQ .NET vloží zprávu pomocí implicitního transakčního programování .NET .

```
Using (TransactionScope scope = new TransactionScope ())
{
    Q.Put (putMsg,pmo);
    scope.Complete ();
}

Q.close();
qMgr.Disconect();}
```

Vysvětlení toku kódu implicitní transakce

Kód vytvoří objekt *TransactionScope* a vloží zprávu do rozsahu. Pak zavolá *Dokončit* , aby informoval koordinátor transakcí o dokončení transakce. Koordinátor transakcí nyní vydá *prepare* a *commit* (potvrdit) k dokončení transakce. Pokud je zjištěn problém, volá se *odvolání* .

Explicitní transakce

Následující kód popisuje, jak aplikace IBM MQ .NET vkládá zprávy pomocí explicitního modelu programování transakcí produktu .NET .

```
MQQueueManager qMgr = new MQQueuemanager ("MQQM);
MQQueue Q = QMGR.AccessQueue("Q", MQC.MQOO_OUTPUT+MQC.MQOO_INPUT_SHARED);
MQPutMessageOptions pmo = new MQPutMessageOptions();
pmo.Options = MQC.MQPMO_SYNCPOINT;
MQMessage putMsg1 = new MQMessage();
```

```
Using(CommittableTransaction tx = new CommittableTransaction()){
Transaction.Current = tx;
    try
    {
    Q.Put(MSG,pmo);
    tx.commit();
    }
    catch(Exception)
    {tx.rollback();}
    }

Q.close();
qMgr.Disconnect();
}
```

Vysvětlení toku kódu explicitní transakce

Část kódu vytvoří transakci pomocí třídy *CommitableTransaction* . Do této oblasti se vloží zpráva a pak při dokončení transakce explicitně volá příkaz *commit* (potvrdit). Jsou-li volány nějaké problémy *rollback* .

Distribuované transakce v nespravovaném režimu

Třídy IBM MQ.NET podporují nespravovaná připojení (klient) s použitím rozšířeného klienta transakcí a COM + /MTS jako koordinátora transakcí pomocí implicitního nebo explicitního modelu programování transakcí. V nespravovaném režimu delegují třídy produktu IBM MQ .NET všechny své volání na klienta rozšířených transakcí jazyka C, který spravuje zpracování transakce jménem produktu .NET.

Zpracování transakce je řízeno externím správcem transakcí a koordinuje globální jednotku práce pod kontrolou rozhraní API správce transakcí. Přísloví MQBEGIN, MQCMIT a MQBACK jsou nedostupné. IBM MQ Třídy .NET odkrývají tuto podporu prostřednictvím svého nespravovaného režimu přenosu (klient jazyka C). Viz téma Konfigurace správců transakcí standardu XA

MTS se vyvinul jako systém zpracování transakcí (TP), který poskytuje stejné funkce jako produkt Windows NT , jak je dostupný v produktu CICS, Tuxedo a na jiných platformách. Je-li nainstalován MTS, do Windows NT se přidá samostatná služba, která se nazývá Microsoft Distributed Transaction Coordinator (MSDTC). MSDTC koordinuje transakce, které zahrnují oddělená datová úložiště nebo prostředky. Pro práci vyžaduje každé datové úložiště pro implementaci svého vlastního proprietárního správce prostředků.

IBM MQ se stane kompatibilní s MSDTC implementací rozhraní (proprietární rozhraní správce prostředků), kde se bude mapovat na volání DTC XA na volání IBM MQ(X/Open). IBM MQ hraje roli správce prostředků.

Když komponenta jako COM + požaduje přístup k IBM MQ, COM obvykle kontroluje odpovídající objekt kontextu MTS, je-li požadována transakce. Je-li požadována transakce, informuje COM diagnostický chybový kód DTC a automaticky spustí integrální transakci IBM MQ pro tuto operaci. Poté bude COM pracovat s daty pomocí softwaru MQMTS, vkládat a dostávat zprávy podle potřeby. Instance objektu získaná z COM volá metodu SetComplete nebo SetAbort poté, co všechny akce na datech jsou u konce. Když aplikace vydá příkaz SetComplete, volání signalizuje kód DTC, že aplikace dokončila transakci a kód DTC může pokračovat v procesu dvoufázového potvrzování. DTC pak vydává volání MQMTS, které při volání na příkaz IBM MQ volají nebo odvolá transakci.

Psaní aplikace IBM MQ .NET pomocí nespravovaného klienta

Pro spuštění v kontextu COM + musí být třída .NET dědit ze systému.EnterpriseServices.ServicedComponent. Pravidla a doporučení pro vytváření sestav, které používají obsluhované komponenty, jsou následující:

Poznámka: Následující kroky jsou relevantní pouze v případě, že používáte režim System.EnterpriseServices .

- Třída a metoda spouštěná v COM + musí být veřejné (žádné vnitřní třídy a žádné chráněné nebo statické metody).
- Atributy třídy a metody: Atribut TransactionOption určuje úroveň transakce třídy, tj. zda jsou transakce zakázány, podporovány nebo vyžadovány. Atribut AutoComplete na metodě ExecuteUOW() instruuje COM +, aby potvrdil transakci, pokud není vyvolána žádná neošetřená výjimka.

- Silné pojmenování sestavení: Sestava musí být v mezipaměti GAC (Global Assembly Cache) pevně pojmenována a registrována. Sestava je registrována v COM + explicitně nebo opožděně zaregistrovaná po registraci v GAC.
- Registrace sestavení v COM +: Příprava sestavení pro vystavení klientům COM. Poté vytvořte knihovnu typů pomocí nástroje pro registraci sestavy, regasm.exe.

```
regasm UnmanagedToManagedXa.dll
```

- Zaregistrujte sestavu na serveru GAC `gacutil /i UnmanagedToManagedXa.dll`.
- Zaregistrujte sestavení v modelu COM + pomocí instalačního nástroje služeb produktu .NET , regsvcs.exe. Prohlédněte si knihovnu typu vytvořenou programem regasm.exe:

```
Regsvcs /appname:UnmanagedToManagedXa /tlb:UnmanagedToManagedXa.tlb  UnmanagedToManagedXa.dll
```

- Sestava je implementována do aplikace GAC a později je registrována v COM + tím, že je zavedena opožděná registrace. Rámec .NET se stará o registraci po prvním spuštění kódu.

Příklad toku kódu pomocí modelu System.EnterpriseServices a System.Transactions s COM + jsou popsány v následujících sekcích:

Příklad toku kódu pomocí modelu System.EnterpriseServices

```
using System;
using IBM.WMQ;
using IBM.WMQ.Nmqi;
using System.Transactions;
using System.EnterpriseServices;

namespace UnmanagedToManagedXa
{

[ComVisible(true)]
[System.EnterpriseServices.Transaction(System.EnterpriseServices.TransactionOption.Required)]
    public class MyXa : System.EnterpriseServices.ServicedComponent
    {

        public MQQueueManager QMGR = null;
        public MQQueueManager QMGR1 = null;
        public MQQueue QUEUE = null;
        public MQQueue QUEUE1 = null;
        public MQPutMessageOptions pmo = null;
        public MQMessage MSG = null;

        public MyXa()
        {
        }

        [System.EnterpriseServices.AutoComplete()]
        public void ExecuteUOW()
        {
            QMGR = new MQQueueManager("usemq");

            QUEUE = QMGR.AccessQueue("SYSTEM.DEFAULT.LOCAL.QUEUE",
                                     MQC.MQOO_INPUT_SHARED +
                                     MQC.MQOO_OUTPUT +
                                     MQC.MQOO_BROWSE);

            pmo = new MQPutMessageOptions();
            pmo.Options = MQC.MQPMO_SYNCPOINT;
            MSG = new MQMessage();
            QUEUE.Put(MSG, pmo);
            QMGR.Disconnect();
        }
    }
}


public void RunNow()
{
   MyXa xa = new MyXa();
   xa.ExecuteUOW();
}
```

Příklad toku kódu pomocí System.Transactions pro interakce s COM +

```
[STAThread]
public void ExecuteUOW()
```

```
{
Hashtable t1 = new Hashtable();
t1.Add(MQC.CHANNEL_PROPERTY, "SYSTEM.DEF.SVRCONN");
t1.Add(MQC.HOST_NAME_PROPERTY, "localhost");
t1.Add(MQC.PORT_PROPERTY, 1414);
t1.Add(MQC.TRANSPORT_PROPERTY, MQC.TRANSPORT_MQSERIES_CLIENT);
TransactionOptions opts = new TransactionOptions();

using(TransactionScope scope = new  TransactionScope(TransactionScopeOption.RequiresNew,
                               opts,     EnterpriseServicesInteropOption.Full)
  {

        QMGR = new MQQueueManager("usemq", t1);
        QUEUE = QMGR.AccessQueue("SYSTEM.DEFAULT.LOCAL.QUEUE",
                                          MQC.MQOO_INPUT_SHARED +
                                          MQC.MQOO_OUTPUT +
                                          MQC.MQOO_BROWSE);

         pmo = new MQPutMessageOptions();
         pmo.Options = MQC.MQPMO_SYNCPOINT;
         MSG = new MQMessage();
         QUEUE.Put(MSG, pmo);
         scope.Complete();
  }
 QMGR.Disconnect();
}
```

Vytváření jednoduchých vložení a získání zpráv v rámci TransactionScope

Ukázkové aplikace C# produktu jsou k dispozici v rámci produktu IBM MQ. Tyto jednoduché aplikace demonstrují vložení a získání zpráv v rámci TransactionScope. Na konci úlohy budete moci vkládat a získávat zprávy z fronty nebo tématu.

Než začnete

Služba MSDTC musí být spuštěna a povolena pro transakce XA.

Informace o této úloze

Příklad je jednoduchou aplikací, `SimpleXAPut` a `SimpleXAGet`. Programy `SimpleXAPut` a `SimpleXAGet` jsou C# aplikace dostupné v rámci IBM MQ. Produkt `SimpleXAPut` demonstruje použití MQPUT v rámci distribuovaných transakcí pomocí oboru názvů SystemTransactions . Produkt `SimpleXAGet` demonstruje použití MQGET v části Distribuované transakce pomocí oboru názvů SystemTransactions .

`SimpleXAPut` se nachází v `WebSphere MQ\tools\dotnet\samples\cs\base`

Postup

Aplikace mohou být spuštěny s parametry příkazového řádku z produktu `tools\dotnet\samples\cs\base\bin` .

```
SimpleXAPut.exe -d destinationURI [-h host -p port -l channel -tx transaction -tm mode -n
numberOfMsgs]
```

```
SimpleXAGet.exe -d destinationURI [-h host -p port -l channel -tx transaction -tm mode -n
numberOfMsgs]
```

kde parametry jsou:

`-destinationURI`

Může jít o frontu nebo téma. Pro frontu zadejte jako `queue://queueName` a pro téma uveďte jako `topic://topicName`.

`-host`

Může se jednat o název hostitele, jako je například localhost nebo adresa IP.

-port

Port, na kterém je spuštěn správce front.

-channel

Kanál připojení, který se používá. Předvolba je SYSTEM.DEF.SVRCONN

-transaction

Výsledek transakce, například potvrzení nebo odvolání transakce.

-mode

Přenosový režim, například spravovaný nebo nespravovaný.

-numberOfMsgs

Počet zpráv. Výchozí hodnota je 1.

Příklad

```
SimpleXAPut -d topic://T01 -h localhost -p 2345 -tx rollback -tm unmanaged
```

```
SimpleXAGet -d queue://Q01 -h localhost -p 2345 -tx rollback -tm unmanaged
```

Obnova transakcí

Tento oddíl popisuje proces obnovení transakcí v produktu IBM MQ .NET XA s použitím spravovaného režimu.

Přehled

Ve zpracování distribuovaných transakcí mohou být transakce úspěšně dokončeny. Mohou však existovat scénáře, ve kterých může transakce selhat z řady důvodů. Mezi tyto důvody může patřit selhání systému, selhání hardwaru, chyba sítě, nesprávná nebo neplatná data, chyby aplikace nebo přírodní nebo člověkem způsobené pohromy. Selhání transakce není možné zabránit selháním transakce. Distribuovaný transakční systém musí být schopen zpracovat tato selhání. Musí být schopen detekovat a opravit chyby, když se vyskytnou. Tento proces je znám jako Zotavení transakcí.

Důležitým aspektem zpracování distribuovaných transakcí je obnova neúplných nebo sporných transakcí. Je nezbytné spustit zotavení, protože část pracovní transakce jednotky práce je zadržena, dokud nebude zotavena. Volba Microsoft.NET ze své knihovny tříd System.Transactions poskytuje volbu pro obnovu neúplných/neověřených transakcí. Tato podpora zotavení očekává, že produkt Resource Manager udržuje protokoly transakcí a v případě potřeby spustí zotavení.

Model obnovení

V modelu zotavení transakce Microsoft .NET , správce transakcí (System.Transactionsnebo koordinátor distribuovaných transakcí Microsoft (MS DTC) nebo obojí), iniciuje, koordinuje a řídí obnovu transakcí. Protokol OLE Tx Protocol (Microsoft XA protocol) poskytuje volby pro konfiguraci DTC k řízení, koordinaci a řízení obnovy pro tyto programy. Aby to bylo možné provést, musí správci prostředků zaregistrovat XA_Switch s MS DTC pomocí nativního rozhraní.

Modul XA_Switch poskytuje vstupní body funkcí XA jako xa_start, xa_end a xa_recover ve správci Resource Manager k koordinátorovi distribuovaných transakcí.

Zotavení pomocí produktu Microsoft Distributed Transaction coordinator (DTC):

Microsoft Koordinátor distribuovaných transakcí poskytuje dva druhy procesů zotavení.

Studené zotavení

Studené zotavení se provádí v případě, že dojde k selhání procesu správce transakcí při otevření připojení ke správci prostředků XA. Po restartování správce transakcí načte protokoly správce transakcí a znovu zavede připojení ke správci prostředků XA a poté zahájí zotavení.

Zotavení po provozu

Obnova za provozu se provádí, pokud správce transakcí zůstane spuštěný, zatímco připojení mezi správcem transakcí a správcem prostředků XA selže, protože selže správce prostředků XA nebo síť. Po selhání se správce transakcí pravidelně pokouší znovu připojit ke správci prostředků XA. Po opětnému zavedení připojení zahájí správce transakcí zotavení XA.

Obor názvů System.Transactions poskytuje spravovanou implementaci distribuovaných transakcí, které jsou založeny na MS DTC jako správce transakcí. Poskytuje podobné funkce jako nativní rozhraní MS DTC, ale v plně spravovaném prostředí. Jediný rozdíl je o obnově transakce. System.Transactions očekává, že správci prostředků budou sami řídit obnovu a pak je koordinovat se správci transakcí (MS DTC). Resource Manager musí požádat o zotavení konkrétní nedokončené transakce a poté ji přijímá a koordinuje na základě skutečného výsledku dané transakce.

Proces zotavení transakce pro IBM MQ .NET
Tento oddíl popisuje, jak mohou být distribuované transakce obnoveny pomocí tříd produktu IBM MQ .NET .

Přehled

Chcete-li obnovit nedokončenou transakci, jsou požadovány informace o obnově. Informace o obnově transakcí musí být protokolovány do úložiště ze strany správců prostředků. IBM MQ Třídy .NET následují po podobné cestě. Informace o obnově transakcí jsou protokolovány do systémové fronty s názvem SYSTEM.DOTNET.XARECOVERY.QUEUE.

Zotavení transakcí v produktu IBM MQ .NET je dvoufázový proces.

1. Protokolování informací o obnově transakcí.
   - Pro každou transakci je během prepare fáze přidána trvalá zpráva obsahující informace o obnově do SYSTEM.DOTNET.XARECOVERY.QUEUE.
   - Pokud je potvrzení úspěšné, zpráva se odstraní.
2. Obnova transakcí pomocí monitorovací aplikace WmqDotnetXAMonitor.
   - WmqDotnetXAMonitor je spravovaná aplikace produktu .NET , která zpracovává zprávy v systému SYSTEM.DOTNET.XARECOVERY.QUEUE a obnovení nedokončených transakcí

   Pokud program MCA nemůže vložit zprávu do cílové fronty, vygeneruje zprávu o výjimce obsahující původní zprávu a vloží ji do přenosové fronty, která má být odeslána do fronty pro odpovědi určené v původní zprávě. (Je-li fronta pro odpovědi ve stejném správci front jako MCA, zpráva se umístí přímo do této fronty, nikoli do přenosové fronty.)

SYSTEM.DOTNET.XARECOVERY.QUEUE

Jedná se o systémovou frontu, která zadržuje informace o obnově transakcí nedokončených transakcí. Tato fronta se vytvoří, když se vytvoří správce front.

Poznámka: Neměli byste odstraňovat SYSTEM.DOTNET.XARECOVERY.QUEUE fronta.

Aplikace WMQDotnetXAMonitor

Komponenta IBM MQ .NET XA Monitor monitoruje daného správce front a obnovuje nedokončené transakce, pokud existují. Následující text se považuje za nedokončené transakce a je obnoven:

Neúplné transakce

- Je-li transakce připravena, ale COMMIT nebyl dokončen během časového limitu.

- Je-li transakce připravena, ale správce front produktu IBM MQ šel dolů.
- Je-li transakce připravena, ale pak se správce transakcí vypnul.

  Monitorovací aplikace musí být spuštěna ze stejného systému, kde běží klientská aplikace IBM MQ .NET . Existují-li aplikace spuštěné ve více systémech, které se připojují ke stejnému správci front, musí být aplikace monitoru spuštěna ze všech systémů. Ačkoli má každý klientský počítač spuštěnou aplikaci monitorování pro obnovení aplikace, každý monitor by měl být schopen identifikovat zprávu, která odpovídá transakci, kterou lokální monitorovací systém MS DTC koordinoval, aby mohl znovu uvést seznam a dokončit jej.

Případy použití zotavení transakce pro IBM MQ .NET
Existuje několik různých případů použití, ze kterých může být třeba transakce obnovit.

- **IBM MQ Aplikace používající jednotlivé instance DTC a jednotlivé instance správce front:** V tomto příkladu použití při připojení ke správci front a spuštění transakce (UoW) v rámci transakce, a pokud se transakce nezdaří a stane se nedokončenou, aplikace Monitor obnoví transakci a dokončí ji.

  V tomto příkladu použití bude spuštěna jediná instance aplikace monitorování, protože k těmto transakcím je přidružen jediný správce front.
- **Několik aplikací produktu IBM MQ používajících jednu instanci DTC a jednu instanci správce front:** V tomto případě existuje více než jedna aplikace WMQ pod jedním DTC a všechny se připojují ke stejnému správci front a spouštějí UoW v rámci transakcí.

  Pokud se transakce nezdaří a stanou se neúplnými, obnoví je aplikace Monitor a dokončí transakce související se všemi aplikacemi.

  V tomto příkladu použití se spustí jedna monitorovací aplikace, protože v transakcích se používá jeden správce front.
- **Více aplikací produktu IBM MQ , více DTC, různé instance správce front:** V tomto příkladu použití existuje více než jedna aplikace WMQ pod různými DTC (tj. každá aplikace běží na jiném počítači) a připojuje se k různým správcům front.

  Pokud dojde k selhání a transakce se stane neúplnou, monitorovací aplikace zkontroluje TransactionManager, kde se nachází ve zprávě, aby určila adresu DTC. Pokud hodnota Whereabouts TransactionManagerodpovídá adrese DTC, pod kterou je monitor spuštěn, dokončí obnovu, jinak pokračuje v hledání, dokud nebude nalezena zpráva odpovídající jejímu DTC.

  V tomto případě bude existovat pouze jedna instance aplikace monitoru běžící na klienta (uživatel nebo počítač), protože každý klient má svého vlastního správce front použitého v transakcích.
- **Více IBM MQ Aplikace, více DTC, více stejných instancí správců front:** V tomto použití existuje více než jedna aplikace WMQ pod různými kódy DTC (každá aplikace běží na jiném počítači) a všechny se připojují ke stejnému správci front.

  Pokud dojde k selhání a transakce se stane neúplnou, monitorovací aplikace ověří TransactionManagerWhereoabouts ve zprávě, aby zkontroloval, zda adresa DTC a hodnota odpovídají DTC, pod kterým je monitor spuštěn. Pokud se obě hodnoty shodují, skončí obnova pokračuje v hledání, dokud nenajde zprávu odpovídající jejímu DTC.

  V tomto příkladu použití bude existovat pouze jediná instance aplikace monitoru na klienta (uživatel nebo počítač), protože každý klient má své vlastní přidružení správce front použité v transakcích.
- **Více aplikací produktu IBM MQ , jednotlivé instance DTC, různé instance správce front:** V tomto případě existuje více než jedna aplikace WMQ pod jedním DTC (tj. na počítači, existuje více než jedna spuštěná aplikace WMQ) a připojuje se k různým správcům front.

  Pokud transakce selže a stane se neúplnou, monitorovací aplikace obnoví transakci.

  V takovém případě bude existovat celá řada instancí aplikace monitorování, ke které se pracuje jako správci front, protože každá aplikace má svého vlastního správce front použitého v transakcích a každý z nich musí být obnoven.

  Poznámka: Není-li aplikace monitoru spuštěna na pozadí, můžete ji spustit.

Použití aplikace WMQDotnetXAMonitor
Aplikace WMQDotnetXAMonitor musí být spuštěna ručně. Může být spuštěn kdykoliv. Můžete jej spustit, když se zobrazí zprávy v systému SYSTEM.DOTNET.XARECOVERY.QUEUE nebo ji můžete nechat běžet na pozadí před tím, než provedete libovolnou transakční práci s aplikacemi, které jsou zapsány pomocí tříd produktu IBM MQ .NET .

Ke spuštění aplikace monitoru použijte následující příkaz

```
WmqDotnetXAMonitor.exe -m QueueManagerName -n ConnectionName -c ChannelName -i
```

Kde

- **-m *QueueManagerNázev***
  Název správce front.

  Volitelné

  -n *ConnectionName*
  Název připojení ve formátu hostitele (port). Název připojení může obsahovat více než jeden název připojení. Více názvů připojení musí být uvedeno v čárkami odděleném seznamu, například `localhost (1414), localhost (1415), localhost (1416)`. Monitor aplikace spustí obnovu pro každý z názvů připojení uvedených v seznamu odděleném čárkami.

  -c *ChannelName*
  Název kanálu.

  -i
  Dokončení heuristického větvení.

  Volitelné

Monitorovací aplikace provádí následující akce:

1. Kontroluje hloubku fronty SYSTEM.DOTNET.XARECOVERY.QUEUE v intervalu 100 sekund.
2. Je-li hloubka fronty větší než nula, monitor XA prochází frontu pro zprávy a kontroluje, zda zpráva odpovídá neúplným kritériím transakce.
3. Pokud některá ze zpráv odpovídá neúplným kritériím transakce, monitor jej stáhne a načte informace o zotavení transakce.
4. Poté určí, zda se informace o obnově vztahují k lokálnímu MS DTC. Pokud ano, pak bude pokračovat v obnově transakce. Jinak se vrátí k procházení další zprávy.
5. Pak zavolá správci front, aby obnovil nedokončenou transakci.

Nastavení konfiguračního souboru aplikace WmqDotNETXAMonitor
Chcete-li monitorovat aplikaci, lze vstupy poskytnout také pomocí konfiguračního souboru aplikace. Ukázkový konfigurační soubor aplikace je dodáván s IBM MQ .NET. Tento soubor může být upraven podle vašich požadavků.

Konfigurační soubor aplikace má nejvyšší prioritu při zvažování vstupních hodnot. Pokud jsou vstupní hodnoty poskytnuty na příkazovém řádku i v konfiguračním souboru aplikace, jsou brány v úvahu i hodnoty z konfigurace aplikace.

Ukázkový konfigurační soubor aplikace.

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
<configSections>
<sectionGroup name="IBM.WMQ">
<section name="dnetxa" type="System.Configuration.NameValueFileSectionHandler"/>
</sectionGroup>
</configSections>
<IBM.WMQ>
<dnetxa>
<add key="ConnectionName" value=""/>
<add key="ChannelName" value=""/>
<add key="QueueManagerName" value=""/>
<add key="UserId" value=""/>
```

```
<add key="SecurityExit" value=""/>
<add key="SecurityExitUserData" value = "">
</dnetxa>
</dnetxa>
</configuration>
```

WmqDotNetXAMonitor
Aplikace monitorování vytvoří soubor protokolu v adresáři aplikace pro protokolování průběhu monitorování a stavu zotavení transakce. Protokolování začíná názvem připojení a podrobnostmi kanálu, aby bylo možné zobrazit aktuálního správce front, pro kterého je zotavení spuštěno.

Jakmile je obnova spuštěna, MessageId zprávy o zotavení transakce, TransactionId nedokončené transakce a skutečný výsledek transakce jako transakce Transaction Manager Coordination bude protokolována.

Ukázkový soubor protokolu:

```
Time|ProcessId|ThreadId|WMQ .NET XA Recovery Monitor, Running now for
ConnectionName:xxxx, Time|ProcessId|ThreadId|Channel=xxxx
Time|ProcessId|ThreadId|Current QueueDepth = n
Time|ProcessId|ThreadId|Current MessageId = xxxx
Time|ProcessId|ThreadId|Current Incomplete Transaction being recovered = xxxxx
Time|ProcessId|ThreadId|Actual Outcome of the transaction(as per DTC)= Commit/Roll back
Time|ProcessId|ThreadId|Recovery Completed for TransactionId= xxxxx
Time|ProcessId|ThreadId|Current QueueDepth = n
Time|ProcessId|ThreadId|Current MessageId = xxxx
Time|ProcessId|ThreadId|Current Incomplete Transaction being recovered = xxxxx
Time|ProcessId|ThreadId|Actual Outcome of the transaction(as per DTC)= Commit/Roll back
Time|ProcessId|ThreadId| Recovery Completed for TransactionId= xxxxx
```

Zápis a implementace programů produktu IBM MQ .NET

Chcete-li použít produkt IBM MQ classes for .NET pro přístup k frontám IBM MQ , naprograte programy v libovolném jazyce, který je podporován produktem .NET a který obsahuje zprávy o vkládání zpráv do front zpráv a získávání zpráv z front produktu IBM MQ .

Dokumentace produktu IBM MQ obsahuje informace pouze v jazycích C#, C++ a Visual Basic.

Tato kolekce témat obsahuje informace, které vám pomohou při psaní aplikací pro interakci se systémy IBM MQ . Podrobnosti o jednotlivých třídách naleznete v tématu Třídy a rozhraní produktu IBM MQ .NET.

Rozdíly spojení

Způsob, jakým program pro produkt IBM MQ.NET obsahuje některé závislosti na režimech připojení, které chcete použít.

Je-li produkt IBM MQ classes for .NET používán jako spravovaný klient, existuje řada rozdílů oproti standardnímu serveru IBM MQ MQI client, protože některé funkce nejsou pro spravovaného klienta dostupné.

Produkt IBM MQ.NET určuje typ připojení, který má být použit při nastavení pro název připojení, název kanálu, hodnotu přizpůsobení NMQ_MQ_LIB a vlastnost MQC.TRANSPORT_PROPERTY.

Připojení spravovaného klienta

Je-li produkt IBM MQ classes for .NET používán jako spravovaný klient, existuje řada rozdílů ze standardní IBM MQ MQI client.

Pro spravovaného klienta nejsou k dispozici následující funkce:

- Komprese kanálu
- Řetězení uživatelských procedur kanálu

Pokusíte-li se tyto funkce používat se spravovaným klientem, bude vrácena výjimka MQException. Je-li chyba zjištěna na straně klienta připojení, použije kód příčiny MQRC_ENVIRONMENT_ERROR. Je-li zjištěn na konci serveru, použije se kód příčiny vrácený serverem.

Uživatelské procedury kanálu zapsané pro nespravovaného klienta nepracují. Musíte zapsat nové uživatelské procedury speciálně pro spravovaného klienta. Zkontrolujte, že v tabulce definic kanálů klienta (CCDT) nejsou zadány žádné platné uživatelské procedury kanálu.

Název uživatelské procedury spravovaného kanálu může mít délku až 999 znaků. Pokud však k určení názvu uživatelské procedury kanálu použijete tabulky CCDT, je tato hodnota omezena na 128 znaků.

Komunikace je podporována pouze přes TCP/IP.

Pokud zastavíte správce front pomocí příkazu **endmqm** , může kanál připojení serveru ke spravovanému klientu produktu .NET trvat delší dobu než kanál připojení serveru k jiným klientům.

Pokud jste nastavili proměnnou *NMQ_MQ_LIB* na hodnotu `managed` , chcete-li používat administrativní diagnostiku problémů IBM MQ , není podporován žádný z parametrů -i, -p, -s, -b nebo -c příkazu **strmqtrc** .

Spravovaná aplikace .NET používající transakce XA nebude pracovat se správcem front z/OS . Spravovaný produkt .NET , který se pokouší připojit ke správci front produktu z/OS , selže s chybou MQRC_UOW_ENLISTMENT_ERROR (mqrc=2354) při volání MQOPEN. Spravovaná aplikace .NET využívající transakce XA však bude pracovat s distribuovaným správcem front.

Definování, který typ připojení se má použít

Typ připojení je určen nastavením názvu připojení, názvu kanálu, hodnoty přizpůsobení NMQ_MQ_LIB a vlastností MQC.TRANSPORT_PROPERTY.

Název připojení můžete zadat takto:

- Explicitně na konstruktoru MQQueueManager :

```
public MQQueueManager(String queueManagerName, MQLONG Options, string Channel,
string ConnName)
```

```
public MQQueueManager(String queueManagerName, string Channel, string ConnName)
```

- Nastavením vlastností MQC.HOST_NAME_PROPERTY a, volitelně, MQC.PORT_PROPERTY v položce hašovací tabulky v konstruktoru MQQueueManager :

```
public MQQueueManager(String queueManagerName, Hashtable properties)
```

- Jako explicitní hodnoty MQEnvironment

```
MQEnvironment.Hostname
```

  `MQEnvironment.Port` (volitelné).
- Nastavením vlastností MQC.HOST_NAME_PROPERTY a, volitelně, MQC.PORT_PROPERTY v hašovací tabulce MQEnvironment.properties .

Název kanálu můžete zadat takto:

- Explicitně na konstruktoru MQQueueManager :

```
public MQQueueManager(String queueManagerName, MQLONG Options, string Channel,
string ConnName)
```

```
public MQQueueManager(String queueManagerName, string Channel, string ConnName)
```

- Nastavením vlastnosti MQC.CHANNEL_PROPERTY se nachází v položce hašovací tabulky v konstruktoru MQQueueManager :

```
public MQQueueManager(String queueManagerName, Hashtable properties)
```

- Jako explicitní hodnota MQEnvironment

```
MQEnvironment.Channel
```

- Nastavením vlastnosti MQC.CHANNEL_PROPERTY v hašovací tabulce MQEnvironment.properties .

Vlastnost přenosu můžete určit tímto způsobem:

- Nastavením vlastnosti MQC.TRANSPORT_PROPERTY v rámci položky hašovací tabulky v konstruktoru MQQueueManager :

```
public MQQueueManager(String queueManagerName, Hashtable properties)
```

- Nastavením vlastnosti MQC.TRANSPORT_PROPERTY v hašovací tabulce MQEnvironment.properties .

Vyberte typ připojení, který požadujete, pomocí jedné z následujících hodnot:

MQC.TRANSPORT_MQSERIES_BINDINGS -připojit se jako server
MQC.TRANSPORT_MQSERIES_CLIENT -připojit se jako klient bez standardu XA
MQC.TRANSPORT_MQSERIES_XACLIENT -připojit se jako klient XA
MQC.TRANSPORT_MQSERIES_MANAGED -připojit se jako spravovaný klient mimo XA

Hodnotu přizpůsobení NMQ_MQ_LIB můžete nastavit tak, aby explicitně vybral typ připojení, jak je uvedeno v následující tabulce.

Hodnota NMQ_MQ_LIB	**Typ připojení**
mqic.dll	Připojit jako klient bez podpory XA
mqicxa.dll	Připojit jako klient XA
mqm.dll	Připojit jako server nebo jako klienta bez podpory XA
Spravováno	Připojit jako spravovaný klient bez podpory XA

Poznámka: Hodnoty proměnné mqic32.dll a mqic32xa.dll jsou při kompatibilitě se staršími verzemi přijímány jako synonyma souborů mqic.dll a mqicxa.dll . Avšak soubor mqm.dll a mqm.pdb jsou součástí balíku klienta pouze od IBM WebSphere MQ 7.1 .

Pokud vyberete typ připojení, který není k dispozici ve vašem prostředí, například zadáte soubor mqic32xa.dll a nemáte podporu XA, produkt IBM MQ.NET vyvolá výjimku.

Nastavení hodnoty NMQ_MQ_LIB na hodnotu "managed" způsobí, že klient bude používat testy diagnostiky problémů se spravovaným produktem IBM MQ , převod dat produktu .NET a další funkce spravovaného nízkoplo- IBM MQ .

Všechny ostatní hodnoty parametru NMQ_MQ_LIB způsobí, že proces .NET použije nespravované diagnostické testy a převod dat produktu IBM MQ a další nespravované funkce IBM MQ (předpokládá se, že je v systému nainstalován produkt IBM MQ MQI client nebo server).

Produkt IBM MQ.NET volí typ připojení tímto způsobem:

1. Pokud je MQC.TRANSPORT_PROPERTY je připojen v souladu s hodnotou proměnné MQC.TRANSPORT_PROPERTY.

   Všimněte si však, že nastavení MQC.TRANSPORT_PROPERTY na MQC.TRANSPORT_MQSERIES_MANAGED nezaručuje, že se proces klienta spustí. I s tímto nastavením není klient spravován v následujících případech:

   - Pokud jiný podproces v procesu navázalo spojení s MQC.TRANSPORT_PROPERTY nastaveno na něco jiného než MQC.TRANSPORT_MQSERIES_MANAGED.

- Pokud není hodnota NMQ_MQ_LIB nastavena na "managed", diagnostické testy problémů, konverze dat a další funkce nízké úrovně nejsou plně spravovány (za předpokladu, že je v systému nainstalován server IBM MQ MQI client nebo server).

2. Pokud byl zadán název připojení bez názvu kanálu nebo byl-li zadán název kanálu bez názvu připojení, bude vrácena chyba.
3. Je-li zadán název připojení i název kanálu, postupujte takto:
   - Je-li hodnota NMQ_MQ_LIB nastavena na hodnotu mqic32xa.dll, připojí se jako klient XA.
   - Je-li hodnota NMQ_MQ_LIB nastavena na spravovaná, připojí se jako spravovaný klient.
   - Jinak se připojí jako klient bez podpory XA.
4. Je-li zadán parametr NMQ_MQ_LIB, připojí se k němu podle hodnoty NMQ_MQ_LIB.
5. Je-li server IBM MQ nainstalován, připojí se jako server.
6. Je-li nainstalován produkt IBM MQ MQI client , připojí se jako klient, který není XA.
7. Jinak se připojí jako spravovaný klient.

Konfigurační soubory pro IBM MQ classes for .NET

Klientská aplikace .NET může použít konfigurační soubor IBM MQ MQI client a, pokud používáte typ spravovaného připojení, konfigurační soubor aplikace .NET . Nastavení v konfiguračním souboru aplikace má prioritu.

Konfigurační soubor klienta

Klientská aplikace produktu IBM MQ classes for .NET může použít konfigurační soubor klienta stejným způsobem jako jiný produkt IBM MQ MQI client. Tento soubor se obvykle nazývá `mqclient.ini`, ale můžete zadat jiný název souboru. Další informace o konfiguračním souboru klienta naleznete v tématu Konfigurace klienta pomocí konfiguračního souboru.

Pouze následující atributy v konfiguračním souboru IBM MQ MQI client jsou relevantní pro IBM MQ classes for .NET. Určíte-li jiné atributy, nemá žádný efekt.

Tabulka 75. Atributy konfiguračního souboru klienta, které jsou relevantní pro IBM MQ classes for .NET

sekce	**Atribut**
kanály	CCSID
kanály	Adresář ChannelDefinition
kanály	ChannelDefinition
kanály	ReconDelay
kanály	DefRecon
kanály	MQReconnectTimeout
kanály	Parametry ServerConnectionParms
kanály	Put1DefaultAlwaysSync
kanály	PasswordProtection
ClientExit-cesta	ExitsDefaultPath
ClientExit-cesta	ExitsDefaultPath64
MessageBuffer	MaximumSize
MessageBuffer	PurgeTime
MessageBuffer	UpdatePercentage

Tabulka 75. Atributy konfiguračního souboru klienta, které jsou relevantní pro IBM MQ classes for .NET (pokračování)	
sekce	**Atribut**
TCP	ClntRcvBufSize
TCP	ClntSndBufSize
TCP	IPAddressVersion
TCP	KeepAlive

Všechny tyto atributy můžete potlačit pomocí příslušné proměnné prostředí.

Konfigurační soubor aplikace

Pracujete-li se spravovaným typem připojení, můžete také přepsat konfigurační soubor klienta IBM MQ a ekvivalentní proměnné prostředí pomocí konfiguračního souboru aplikace .NET .

Nastavení konfiguračního souboru aplikace .NET se zpracovává pouze při spuštění s typem spravovaného připojení a jsou ignorovány pro jiné typy připojení.

Konfigurační soubor aplikace .NET a jeho formát jsou definovány produktem Microsoft pro obecné použití v rámci struktury .NET , ale jednotlivé názvy sekcí, klíče a hodnoty uvedené v této dokumentaci jsou specifické pro produkt IBM MQ.

Formát konfiguračního souboru aplikace .NET je číslo *oddílů*. Každá sekce obsahuje jeden nebo více *klíčů*a každý klíč má přidruženou *hodnotu*. Následující příklad zobrazuje sekce, klíče a hodnoty použité v konfiguračním souboru aplikace .NET pro řízení vlastnosti **TCP/IP KeepAlive** :

```
<configuration>
  <configSections>
    <section name="TCP" type="System.Configuration.NameValueSectionHandler"/>
  </configSections>
  <TCP>
    <add key="KeepAlive" value="true"></add>
  </TCP>
<configuration>
```

Klíčová slova použitá v názvech sekcí a klíčů konfiguračního souboru aplikace .NET se přesně shodují s klíčovými slovy pro oddíly a atributy definované v konfiguračním souboru klienta.

Sekce `<configSections>` musí být prvním podřízeným prvkem prvku `<configuration>` .

Další informace naleznete v dokumentaci produktu Microsoft .

Příklad fragmentu kódu C# pro použití s .NET

Fragment kódu C# demonstrující, že se aplikace připojuje ke správci front, vloží zprávu do fronty a obdrží odpověď.

Následující fragment kódu C# demonstruje aplikaci, která provádí tři akce:

1. Připojit se ke správci front
2. Vložte zprávu do systému SYSTEM.DEFAULT.LOCAL.QUEUE
3. Získejte zprávu zpět

Také ukazuje, jak změnit typ připojení.

```
// ============================================================================
// Licensed Materials - Property of IBM
// 5724-H72
// (c) Copyright IBM Corp. 2003, 2023
// ============================================================================
using System;
```

```
using System.Collections;

using IBM.WMQ;

class MQSample
{
  // The type of connection to use, this can be:-
  // MQC.TRANSPORT_MQSERIES_BINDINGS for a server connection.
  // MQC.TRANSPORT_MQSERIES_CLIENT for a non-XA client connection
  // MQC.TRANSPORT_MQSERIES_XACLIENT for an XA client connection
  // MQC.TRANSPORT_MQSERIES_MANAGED for a managed client connection
  const String connectionType = MQC.TRANSPORT_MQSERIES_CLIENT;

  // Define the name of the queue manager to use (applies to all connections)
  const String qManager = "your_Q_manager";

  // Define the name of your host connection (applies to client connections only)
  const String hostName = "your_hostname";

  // Define the name of the channel to use (applies to client connections only)
  const String channel = "your_channelname";



  /// <summary>
  /// Initialise the connection properties for the connection type requested
  /// </summary>
  /// <param name="connectionType">One of the MQC.TRANSPORT_MQSERIES_ values</param>
  static Hashtable init(String connectionType)
  {
    Hashtable connectionProperties = new Hashtable();

    // Add the connection type
    connectionProperties.Add(MQC.TRANSPORT_PROPERTY, connectionType);

    // Set up the rest of the connection properties, based on the
    // connection type requested
    switch(connectionType)
    {
      case MQC.TRANSPORT_MQSERIES_BINDINGS:
        break;
      case MQC.TRANSPORT_MQSERIES_CLIENT:
      case MQC.TRANSPORT_MQSERIES_XACLIENT:
      case MQC.TRANSPORT_MQSERIES_MANAGED:
        connectionProperties.Add(MQC.HOST_NAME_PROPERTY, hostName);
        connectionProperties.Add(MQC.CHANNEL_PROPERTY, channel);
        break;
    }

    return connectionProperties;
  }
  /// <summary>
  /// The main entry point for the application.
  /// </summary>
  [STAThread]
  static int Main(string[] args)
  {
    try
    {
      Hashtable connectionProperties = init(connectionType);

      // Create a connection to the queue manager using the connection
      // properties just defined
      MQQueueManager qMgr = new MQQueueManager(qManager, connectionProperties);

      // Set up the options on the queue we want to open
      int openOptions = MQC.MQOO_INPUT_AS_Q_DEF | MQC.MQOO_OUTPUT;

      // Now specify the queue that we want to open,and the open options
      MQQueue system_default_local_queue =
        qMgr.AccessQueue("SYSTEM.DEFAULT.LOCAL.QUEUE", openOptions);

      // Define an IBM MQ message, writing some text in UTF format
      MQMessage hello_world = new MQMessage();
      hello_world.WriteUTF("Hello World!");

      // Specify the message options
      MQPutMessageOptions pmo = new MQPutMessageOptions(); // accept the defaults,
                                                           // same as MQPMO_DEFAULT

      // Put the message on the queue
      system_default_local_queue.Put(hello_world, pmo);
```

```
      // Get the message back again

      // First define an IBM MQ message buffer to receive the message
      MQMessage retrievedMessage =new MQMessage();
      retrievedMessage.MessageId =hello_world.MessageId;

      // Set the get message options
      MQGetMessageOptions gmo =new MQGetMessageOptions(); //accept the defaults
                                                          //same as MQGMO_DEFAULT

      // Get the message off the queue
      system_default_local_queue.Get(retrievedMessage,gmo);

      // Prove we have the message by displaying the UTF message text
      String msgText = retrievedMessage.ReadUTF();
      Console.WriteLine("The message is: {0}", msgText);

      // Close the queue
      system_default_local_queue.Close();

      // Disconnect from the queue manager
      qMgr.Disconnect();
    }

    //If an error has occurred,try to identify what went wrong.

    //Was it an IBM MQ error?
    catch (MQException ex)
    {
      Console.WriteLine("An IBM MQ error occurred: {0}", ex.ToString());
    }

    catch (System.Exception ex)
    {
      Console.WriteLine("A System error occurred: {0}", ex.ToString());
    }

    return 0;
  }//end of start
}//end of sample
```

Nastavení prostředí produktu IBM MQ

Než použijete připojení klienta k připojení ke správci front, je třeba nastavit prostředí produktu IBM MQ .

Poznámka: Tento krok není nutný při použití produktu IBM MQ classes for .NET v režimu vazeb serveru.

Programovací rozhraní .NET umožňuje použít hodnotu přizpůsobení NMQ_MQ_LIB, ale také zahrnuje třídu MQEnvironment. Tato třída vám umožňuje uvést podrobnosti, které se mají použít během pokusu o připojení, jako například v následujícím seznamu:

- Název kanálu
- Název hostitele
- Číslo portu
- Uživatelské procedury kanálu
- Parametry zabezpečení SSL
- ID uživatele a heslo

Úplné informace o třídě MQEnvironment naleznete v tématu Třída MQEnvironment.NET .

Chcete-li určit název kanálu a název hostitele, použijte následující kód:

```
MQEnvironment.Hostname = "host.domain.com";
MQEnvironment.Channel  = "client.channel";
```

Při výchozím nastavení se klienti pokusí připojit k modulu listener produktu IBM MQ na portu 1414. Chcete-li určit jiný port, použijte tento kód:

```
MQEnvironment.Port = nnnn;
```

Připojování k správci front a odpojování od něj

Pokud jste nakonfigurovali prostředí produktu IBM MQ , jste připraveni připojit se ke správci front.

Chcete-li se připojit ke správci front, vytvořte novou instanci třídy MQQueueManager :

```
MQQueueManager queueManager = new MQQueueManager("qMgrName");
```

Chcete-li se odpojit od správce front, volejte metodu `Disconnect` ve správci front:

```
queueManager.Disconnect();
```

Chcete-li se připojit ke správci front, musíte mít při pokusu o připojení ke správci front oprávnění k dotazům (`inq`). Bez dotazovacího oprávnění se pokus o připojení nezdaří.

Vočíte-li metodu `Disconnect` , všechny otevřené fronty a procesy, ke kterým jste přistoupil prostřednictvím tohoto správce front, budou zavřeny. Je však dobrým programovacím postupem, abyste tyto prostředky při jejich používání explicitně uzavřeli. Chcete-li zavřít prostředky, použijte metodu `Close` na objektu přidruženém ke každému prostředku.

Metody `Commit` a `Backout` ve správci front nahradí volání MQCMIT a MQBACK, která se používají spolu s procedurálním rozhraním.

Přístup k frontám a tématům

K frontám a tématům můžete přistupovat pomocí metod produktu MQQueueManager nebo příslušných konstruktorů.

Chcete-li přistupovat k frontám, použijte metody třídy MQQueueManager . MQOD (struktura deskriptoru objektu) je sbalena do parametrů těchto metod. Chcete-li například otevřít frontu ve správci front představovaném objektem MQQueueManager s názvem queueManager, použijte následující kód:

```
MQQueue queue = queueManager.AccessQueue("qName",
                                         MQC.MQOO_OUTPUT,
                                         "qMgrName",
                                         "dynamicQName",
                                         "altUserId");
```

Parametr *options* je stejný jako parametr Options v rámci volání MQOPEN.

Metoda AccessQueue vrací nový objekt třídy MQQueue.

Až skončíte s používáním fronty, použijte metodu Close () k zavření, jako v následujícím příkladu:

```
queue.Close();
```

V produktu IBM MQ .NETmůžete také vytvořit frontu pomocí konstruktoru MQQueue. Parametry jsou přesně stejné jako u metody accessQueue spolu s přidáním parametru správce front, který určuje instanci objektu MQQueueManager , jež má být použita. Příklad:

```
MQQueue queue = new MQQueue(queueManager,
                            "qName",
                            MQC.MQOO_OUTPUT,
                            "qMgrName",
                            "dynamicQName",
                            "altUserId");
```

Při vytváření objektu fronty tímto způsobem lze zapisovat do vlastních podtříd fronty MQQueue.

Podobně můžete také přistupovat k tématům s použitím metod třídy MQQueueManager . Chcete-li otevřít téma, použijte metodu AccessTopic(). Tento příkaz vrátí nový objekt třídy MQTopic. Po dokončení práce s tématem použijte metodu Close () objektu MQTopic k jeho zavření.

Také můžete vytvořit téma pomocí konstruktoru MQTopic. Pro témata existuje řada konstruktorů; další informace viz MQTopic.NET class.

Práce se zprávami

Zprávy se zpracovávají pomocí metod fronty nebo tříd témat. Chcete-li vytvořit novou zprávu, vytvořte nový objekt MQMessageobject.

Vložení zpráv do front nebo témat pomocí metody Put () třídy MQQueue nebo MQTopic. Načtení zpráv z front nebo témat pomocí metody Get () třídy MQQueue nebo MQTopic. Na rozdíl od procedurálního rozhraní, kde příkazy MQPUT a MQGET vložit a získat pole bajtů, IBM MQ classes for .NET put and get instances of the MQMessage class. Třída MQMessage zapouzdřuje vyrovnávací paměť dat, která obsahuje skutečná data zprávy, spolu se všemi parametry MQMD (Message Descriptor), které popisují danou zprávu.

Chcete-li vytvořit novou zprávu, vytvořte novou instanci třídy MQMessage a pomocí metod WriteXXX umístěte data do vyrovnávací paměti zpráv.

Když se vytvoří nová instance zprávy, všechny parametry MQMD se automaticky nastaví na jejich výchozí hodnoty, jak je definováno v Počáteční hodnoty a deklarace jazyka pro MQMD. Metoda PUT () fronty MQQueue také vezme instanci třídy voleb MQPutMessagejako parametr. Tato třída představuje strukturu MQPMO. Následující příklad vytvoří zprávu a vloží ji do fronty:

```
// Build a new message containing my age followed by my name
MQMessage myMessage = new MQMessage();
myMessage.WriteInt(25);

String name = "Charlie Jordan";
myMessage.WriteUTF(name);

// Use the default put message options...
MQPutMessageOptions pmo = new MQPutMessageOptions();

// put the message
!queue.Put(myMessage,pmo);
```

Metoda Get () MQQueue vrací novou instanci MQMessage, která představuje zprávu právě převzaté z fronty. Jako parametr má také instanci třídy voleb MQGetMessage. Tato třída představuje strukturu MQGMO.

Maximální velikost zprávy není třeba zadávat, protože metoda Get () automaticky upravuje velikost vnitřní vyrovnávací paměti tak, aby odpovídala příchozí zprávě. Použijte metody ReadXXX třídy MQMessage pro přístup k datům ve vrácené zprávě.

Následující příklad uvádí, jak získat zprávu z fronty:

```
// Get a message from the queue
MQMessage theMessage    = new MQMessage();
MQGetMessageOptions gmo = new MQGetMessageOptions();
queue.Get(theMessage,gmo);  // has default values

// Extract the message data
int age = theMessage.ReadInt();
String name1 = theMessage.ReadUTF();
```

Formát čísla, který metody čtení a zápisu používají, můžete změnit nastavením proměnné člena *encoding* .

Můžete změnit znakovou sadu, která má být použita pro čtení a zápis řetězců, nastavením proměnné členu *characterSet* .

Viz MQMessage.NET třída , kde získáte další podrobnosti.

Poznámka: Metoda WriteUTF() MQMessage automaticky kóduje délku řetězce stejně jako bajty Unicode, které obsahuje. Když bude vaše zpráva přečtena jiným programem .NET (pomocí ReadUTF()), je to nejjednodušší způsob, jak odeslat informace o řetězci.

Práce s vlastnostmi zprávy

Vlastnosti zpráv umožňují vybírat zprávy nebo načítat informace o zprávě bez přístupu k jejím záhlavím. Třída MQMessage obsahuje metody pro získání a nastavení vlastností.

Vlastnosti zpráv můžete použít k povolení výběru zpráv pro zpracování nebo načítání informací o zprávě bez přístupu k záhlavím MQMD nebo MQRFH2 . Usnadňují také komunikaci mezi aplikacemi IBM MQ a JMS . Další informace o vlastnostech zprávy v produktu IBM MQnaleznete v tématu Vlastnosti zprávy.

Třída MQMessage poskytuje řadu metod pro získání a nastavení vlastností v závislosti na typu dat vlastnosti. Metody get mají názvy formátu Get * Property a metody set mají názvy z formátu Set * Property, kde hvězdička (*) představuje jeden z následujících řetězců:

- Logická hodnota
- bajt
- Bajtů
- Dvojitá čára
- Pohyblivá desetinná čárka
- Int
- Int2
- Int4
- Int8
- Dlouhý
- Objekt
- Krátký
- Řetězec

Chcete-li například získat vlastnost IBM MQ myproperty (znakový řetězec), použijte volání `message.GetStringProperty('myproperty')`. Volitelně můžete předat deskriptor vlastnosti, který IBM MQ bude dokončen.

Ošetření chyb

Zpracovávat chyby vznikající z produktu IBM MQ classes for .NET pomocí bloků `try` a `catch` .

Metody v rozhraní produktu .NET nevracejí kód dokončení a kód příčiny. Místo toho vyhodí výjimku pokaždé, když kód dokončení a kód příčiny, které jsou výsledkem volání IBM MQ , nejsou obě nula. To zjednodušuje logiku programu, takže nemusíte kontrolovat návratové kódy po každém volání do IBM MQ. Můžete se rozhodnout, ve kterých bodech ve vašem programu se chcete vypořádat s možností selhání. V těchto bodech můžete obklopovat kód pomocí bloků `try` a `catch` , jako v následujícím příkladu:

```
try
{
  myQueue.Put(messageA,PutMessageOptionsA);
  myQueue.Put(messageB,PutMessageOptionsB);
}
catch (MQException ex)
{
  // This block of code is only executed if one of
  // the two put methods gave rise to a non-zero
  // completion code or reason code.
  Console.WriteLine("An error occurred during the put operation:" +
                        "CC = " + ex.CompletionCode +
                        "RC = " + ex.ReasonCode);
  Console.WriteLine("Cause exception:" + ex );
}
```

Získání a nastavení hodnot atributu

Třídy MQManagedObject, MQQueue a MQQueueManager obsahují metody, které vám umožňují získat a nastavit hodnoty atributu. Všimněte si, že metody pro frontu MQQueue pracují pouze v případě, že při otevření fronty zadáte příslušné parametry dotazu a nastavení.

Pro obecné atributy dědí třídy MQQueueManager a MQQueue ze třídy s názvem MQManagedObject. Tato třída definuje rozhraní Inquire () a Set ().

Když vytváříte nový objekt správce front pomocí operátoru *new* , je automaticky otevřen pro zjišťování. Použijete-li metodu AccessQueue() k přístupu k objektu fronty, tento objekt se automaticky neotevře pro dotaz nebo operaci nastavení, což by mohlo způsobit problémy s některými typy vzdálených front. Chcete-li použít metody Inquire and Set a nastavit vlastnosti ve frontě, je třeba v parametru openOptions metody AccessQueue() určit příslušné parametry dotazu a nastavit příslušné parametry.

Metody dotazování a nastavení mají tři parametry:

- pole selektorů
- Pole intAttrs
- pole charAttrs

Nepotřebujete parametry SelectorCount, IntAttrCount a CharAttrLength, které se nacházejí v MQINQ, protože délka pole je vždy známá. Následující příklad uvádí, jak provést dotaz na frontu:

```
//inquire on a queue
int [ ] selectors = new int [2] ;
int [ ] intAttrs = new int [1] ;
byte [ ] charAttrs = new byte [MQC.MQ_Q_DESC_LENGTH];
selectors [0] = MQC.MQIA_DEF_PRIORITY;
selectors [1] = MQC.MQCA_Q_DESC;
queue.Inquire(selectors,intAttrs,charAttrs);
ASCIIEncoding enc = new ASCIIEncoding();
String s1 = "";
s1 = enc.GetString(charAttrs);
```

Všechny atributy těchto objektů mohou být dotazovány. Podmnožina atributů je vystavena jako vlastnosti objektu. Seznam atributů objektů najdete v tématu Atributy objektů. Pro vlastnosti objektu si prohlédněte odpovídající popis třídy.

Vícevláknové programy

Běhové prostředí produktu .NET je ve své podstatě vícevláknové. Produkt IBM MQ classes for .NET umožňuje sdílení objektu správce front ve více podprocesech, ale zajišťuje synchronizaci všech přístupů k cílovému správci front.

Zvažte jednoduchý program, který se připojí ke správci front a otevře frontu při spuštění. Program zobrazí na obrazovce jediné tlačítko. Když uživatel klepne na toto tlačítko, program načte zprávu z fronty. V této situaci se inicializace aplikace vyskytuje v jednom podprocesu a kód, který se provádí v odpovědi na stisknutí tlačítka, se provádí v samostatném podprocesu (podproces uživatelského rozhraní).

Implementace produktu IBM MQ .NET zajišťuje, že pro konkrétní připojení (instance objektuMQQueueManager) je veškerý přístup k cílovému správci front IBM MQ synchronizován. Výchozí chování je, že podproces, který chce vydat volání správci front, je blokován, dokud nejsou dokončena všechna ostatní volání v průběhu tohoto připojení. Pokud vyžadujete souběžný přístup ke stejnému správci front z více podprocesů ve svém programu, vytvořte nový objekt MQQueueManager pro každý podproces, který vyžaduje souběžný přístup. (Toto je ekvivalent k zadání samostatného volání MQCONN pro každé vlákno.)

Pokud jsou výchozí volby připojení přepsány MQC.MQCNO_HANDLE_SHARE_NONE nebo MQC.MQCNO_SHARE_NO_BLOCK , poté správce front již není synchronizován.

Použití tabulky definic kanálů klienta s .NET

Pro produkt IBM MQlze použít tabulku CCDT (Client Channel Definition table) se třídami .NET . Umístění tabulky CCDT určujete různými způsoby v závislosti na tom, zda používáte spravované nebo nespravované připojení.

Typ nespravovaného připojení klienta bez podpory XA nebo XA

V případě typu nespravovaného připojení můžete určit umístění tabulky CCDT dvěma způsoby:

- Pomocí proměnných prostředí MQCHLLIB určete adresář, kde se tabulka nachází, a MQCHLTAB pro určení názvu souboru tabulky.
- Použije se konfigurační soubor klienta. Ve stanze CHANNELS použijte atributy ChannelDefinitionAdresář k uvedení adresáře, kde se tabulka nachází, a ChannelDefinitionSoubor pro uvedení názvu souboru.

Je-li umístění určeno v konfiguračním souboru klienta i pomocí proměnných prostředí, proměnné prostředí budou mít prioritu. Tuto funkci můžete použít k určení standardního umístění v konfiguračním souboru klienta a k přepsání pomocí proměnných prostředí, je-li to nutné.

Typ připojení spravovaného klienta

Se spravovaným typem připojení můžete určit umístění tabulky CCDT třemi způsoby:

- Pomocí konfiguračního souboru aplikace .NET . V sekci CHANNELS použijte klíče ChannelDefinitionAdresář k určení adresáře, kde je tabulka umístěna, a ChannelDefinitionSoubor pro uvedení názvu souboru.
- Pomocí proměnných prostředí MQCHLLIB určete adresář, kde se tabulka nachází, a MQCHLTAB pro určení názvu souboru tabulky.
- Použije se konfigurační soubor klienta. Ve stanze CHANNELS použijte atributy ChannelDefinitionAdresář k uvedení adresáře, kde se tabulka nachází, a ChannelDefinitionSoubor pro uvedení názvu souboru.

Je-li umístění uvedeno více než jedním z těchto způsobů, proměnné prostředí mají přednost před konfiguračním souborem klienta a konfigurační soubor aplikace .NET má přednost před ostatními metodami. Tuto funkci můžete použít k určení standardního umístění v konfiguračním souboru klienta a přepsání nastavení pomocí proměnných prostředí nebo konfiguračního souboru aplikace, je-li to nutné.

Jak aplikace .NET určuje, jaká definice kanálu se má použít

V prostředí klienta produktu IBM MQ .NET může být definice kanálu, která má být použita, zadána různými způsoby. Může existovat více specifikací definice kanálu. Aplikace odvozuje definici kanálu z jednoho nebo více zdrojů.

Pokud existuje více než jedna definice kanálu, vybere se jedna z nich v následujícím pořadí priority:

1. Vlastnosti určené v konstruktoru MQQueueManager , ať už explicitně, nebo prostřednictvím zahrnutí *MQC.CHANNEL_PROPERTY* v tabulce hashtable vlastností
2. Vlastnost *MQC.CHANNEL_PROPERTY* v hašovací tabulce MQEnvironment.properties .
3. Vlastnost *Kanál* v MQEnvironment
4. Konfigurační soubor aplikace .NET , název sekce CHANNELS, klíč ServerConnectionParms (používá se pouze pro spravovaná připojení)
5. Proměnná prostředí *MQSERVER*
6. Konfigurační soubor klienta, stanza CHANNELS, Atribut ServerConnectionParms
7. Tabulka definic kanálů klienta (CCDT). Umístění tabulky CCDT je určeno v konfiguračním souboru aplikace .NET (vztahuje se pouze na spravovaná připojení).
8. Tabulka definic kanálů klienta (CCDT). Umístění tabulky CCDT se zadává pomocí proměnných prostředí *MQCHLIB* a *MQCHLTAB* .
9. Tabulka definic kanálů klienta (CCDT). Umístění tabulky CCDT je určeno pomocí konfiguračního souboru klienta

U položek 1-3 je definice kanálu sestavena pole podle pole z hodnot poskytnutých aplikací. Tyto hodnoty mohou být poskytnuty pomocí různých rozhraní a pro každou z nich může existovat více hodnot. Hodnoty polí se přidávají do definice kanálu podle zadaného pořadí priority:

1. Hodnota položky *connName* v konstruktoru MQQueueManager
2. Hodnoty vlastností ze hašovací tabulky MQQueueManager.properties
3. Hodnoty vlastností hašovací tabulky MQEnvironment.properties
4. Hodnoty nastavené jako pole MQEnvironment (například MQEnvironment.Hostname, MQEnvironment.Port)

U položek 4-6 je jako hodnota dodána celá definice kanálu. Neurčená pole v definici kanálu mají výchozí nastavení systému. Žádné hodnoty z jiných metod definování kanálů a jejich polí se neslučují s těmito specifikacemi.

U položek 7-9 je celá definice kanálu převzata z tabulky CCDT. Pole, která nebyla výslovně uvedena, když byl kanál definován, mají výchozí nastavení systému. Žádné hodnoty z jiných metod definování kanálů a jejich polí se neslučují s těmito specifikacemi.

Použití kanálů kanálů v produktu IBM MQ .NET

Pokud používáte vazby klienta, můžete použít uživatelské procedury kanálu jako pro jakékoli jiné připojení klienta. Používáte-li spravované vazby, musíte napsat uživatelský program, který implementuje příslušné rozhraní.

Vazby klienta

Pokud používáte vazby klienta, můžete použít uživatelské procedury kanálu, jak je popsáno v tématu Uživatelské procedury kanálu. Nelze použít uživatelské procedury kanálu zapsané pro spravované vazby.

Spravované vazby

Používáte-li spravované připojení k implementaci uživatelské procedury, nadefinujete novou třídu .NET , která implementuje příslušné rozhraní. V balíku IBM MQ jsou definována tříuživatelská rozhraní:

- MQSendExit
- MQReceiveExit
- MQSecurityExit

Poznámka: Uživatelské procedury zapsané pomocí těchto rozhraní nejsou v nespravovaném prostředí podporovány jako uživatelské procedury kanálu.

Následující ukázka definuje třídu, která implementuje všechny tři:

```
class MyMQExits : MQSendExit, MQReceiveExit, MQSecurityExit
{
   // This method comes from the send exit
  byte[] SendExit(MQChannelExit       channelExitParms,
                  MQChannelDefinition channelDefinition,
                  byte[]              dataBuffer
                  ref int             dataOffset
                  ref int             dataLength
                  ref int             dataMaxLength)
  {
     // complete the body of the send exit here
  }


  // This method comes from the receive exit
  byte[] ReceiveExit(MQChannelExit       channelExitParms,
                     MQChannelDefinition channelDefinition,
                     byte[]              dataBuffer
                     ref int             dataOffset
                     ref int             dataLength
                     ref int             dataMaxLength)
  {
     // complete the body of the receive exit here
```

```
  }


  // This method comes from the security exit
  byte[] SecurityExit(MQChannelExit       channelExitParms,
                      MQChannelDefinition channelDefParms,
                      byte[]              dataBuffer
                      ref int             dataOffset
                      ref int             dataLength
                      ref int             dataMaxLength)
  {
     // complete the body of the security exit here
  }

}
```

Každé uživatelské proceduře je předána funkce MQChannelExit a instance objektu MQChannelDefinition . Tyto objekty reprezentují struktury MQCXP a MQCD definované v procedurálním rozhraní.

Data, která mají být odeslána uživatelskou procedurou pro odeslání zprávy, a data přijatá v rámci zabezpečení nebo procedury příjmu, jsou určena pomocí parametrů uživatelské procedury.

U položky data na offsetu *dataOffset* s délkou *dataLength* v bajtovém poli *dataBuffer* jsou data, která mají být odeslána uživatelskou procedurou odeslání, a data přijatá v rámci zabezpečení nebo při ukončení příjmu. Parametr *dataMaxLength* poskytuje maximální délku (from *dataOffset*). k dispozici pro uživatelskou proceduru v umístění *dataBuffer*. Poznámka: Pro uživatelskou proceduru zabezpečení je možné, aby dataBuffer měla hodnotu null, pokud se jedná o první zavolání ukončení procedury nebo ukončení partnera, který má odeslat žádná data.

Na oplátku by hodnota *dataOffset* a *dataLength* měla být nastavena tak, aby ukazovala na posun a délku v rámci vráceného bajtového pole, které by měly třídy .NET používat. Pro uživatelskou proceduru odeslání to označuje data, která má odeslat, a pro uživatelskou proceduru pro zabezpečení nebo příjem dat, která by měla být interpretována. Ukončení by mělo normálně vrátit bajtové pole; výjimky jsou procedury zabezpečení, které se mohou rozhodnout neodesílat žádná data, a všechny uživatelské procedury volané s příčinami INIT nebo TERM. Nejjednodušší způsob výstupu, který lze zapsat, je tedy takový, který nedělá nic jiného než návrat dataBuffer:

Nejjednodušším možným výstupním tělem je:

```
{
  return dataBuffer;
}
```

Třída MQChannelDefinition

ID uživatele a heslo určené pomocí spravované aplikace klienta .NET jsou nastaveny ve třídě IBM MQ .NET MQChannelDefinition , která je předána uživatelské proceduře zabezpečení klienta. Uživatelská procedura zabezpečení zkopíruje ID uživatele a heslo na disk MQCD.RemoteUserIdentifier a MQCD.RemotePassword (viz “Psaní uživatelské procedury zabezpečení” na stránce 939).

Určení uživatelských procedur kanálu (spravovaného klienta)

Zadáte-li při vytváření objektu MQQueueManager název kanálu a název připojení (buď v konstruktoru MQEnvironment nebo v konstruktoru MQQueueManager), můžete kanály kanálu zadat dvěma způsoby.

V pořadí přednosti jsou tyto:

1. Předávání vlastností hašovací tabulky MQC.SECURITY_EXIT_PROPERTY, MQC.SEND_EXIT_PROPERTY nebo MQC.RECEIVE_EXIT_PROPERTY v konstruktoru MQQueueManager .
2. Nastavení vlastností MQEnvironment SecurityExit, SendExit nebo ReceiveExit .

Nezadáte-li název kanálu a název připojení, budou kanály kanálu používané k použití pocházející z definice kanálu převzaté z tabulky definic kanálů klienta (CCDT). Není možné přepsat hodnoty uložené v definici kanálu. Další informace o tabulkách definic kanálů naleznete v tématu Tabulka definic kanálů klienta a “Použití tabulky definic kanálů klienta s .NET” na stránce 540 .

V každém případě má specifikace tvar řetězce s následujícím formátem:

```
Assembly_name(Class_name)
```

Jméno_třídy je úplný název, včetně specifikace oboru názvů, třídy .NET , která implementuje IBM.WMQ.MQSecurityExit, IBM.WMQ.MQSendExit nebo IBM.WMQ.MQReceiveExit (podle potřeby). *Assembly_name* je plně kvalifikované umístění sestavy, včetně přípony souboru, sestavy, která obsahuje třídu. Délka řetězce je omezena na 999 znaků, používáte-li vlastnosti prostředí MQEnvironment nebo MQQueueManager. Je-li však název uživatelské procedury kanálu zadán v tabulce CCDT, je omezen na 128 znaků. Je-li to nezbytné, načte kód klienta produktu .NET a vytvoří instanci zadané třídy analýzou specifikace řetězce.

Určení uživatelských dat uživatelské procedury kanálu (spravovaného klienta)

Uživatelské procedury kanálu mohou mít k sobě přidružená uživatelská data. Pokud při vytváření objektu MQQueueManager zadáte název kanálu a název připojení (buď v konstruktoru MQEnvironment, nebo v konstruktoru MQQueueManager), můžete data uživatele zadávat dvěma způsoby.

V pořadí přednosti jsou tyto:

1. Předávání vlastností hašovací tabulky MQC.SECURITY_USERDATA_PROPERTY, MQC.SEND_USERDATA_PROPERTY nebo MQC.RECEIVE_USERDATA_PROPERTY v konstruktoru MQQueueManager .
2. Nastavení vlastností dat SecurityUserdat MQEnvironment, SendUserData nebo ReceiveUser.

Pokud nezadáte název kanálu a název připojení, použijí se hodnoty výstupních uživatelských dat, které mají být použity, pocházející z definice kanálu z tabulky CCDT (Client Channel Definition table). Není možné přepsat hodnoty uložené v definici kanálu. Další informace o tabulkách definic kanálů naleznete v tématu Tabulka definic kanálů klienta a “Použití tabulky definic kanálů klienta s .NET” na stránce 540 .

V každém případě je specifikace řetězec, omezen na 32 znaků.

Automatické opětovné připojení klienta v .NET

Můžete nastavit, aby se váš klient znovu automaticky připojil ke správci front během neočekávaného přerušení spojení.

Klient může být neočekávaně odpojen od správce front, pokud se například zastaví správce front nebo selže síť nebo server.

Bez automatického opětovného připojení klienta dojde k chybě při selhání připojení. Můžete použít kód chyby, který vám pomůže znovu ustanovit spojení.

Klient, který používá poskytovanou službu automatického připojení klienta, se nazývá znovu připojitelného klienta. Chcete-li vytvořit znovu připojitelného klienta, uveďte určité volby, které se nazývají volby opětovného připojení při připojování ke správci front.

Je-li klientská aplikace klientem produktu IBM MQ .NET , může se rozhodnout získat automatické opětovné připojení klienta zadáním příslušné hodnoty pro CONNECT_OPTIONS_PROPERTY, když použijete třídu `MQQueueManager` k vytvoření správce front. Podrobné informace o hodnotách CONNECT_OPTIONS_PROPERTY naleznete v tématu Volby opětovného připojení .

Můžete vybrat, zda se klientská aplikace vždy připojí a znovu připojí ke správci front stejného názvu, ke stejnému správci front nebo k jakékoli sadě správců front, které jsou definovány se stejným parametrem `QMNAME` v tabulce připojení klienta (podrobnosti viz Skupiny správců front v CCDT).

Podpora TLS (Transport Layer Security) pro .NET

Klientské aplikace produktu IBM MQ classes for .NET podporují šifrování TLS (Transport Layer Security). Protokol TLS poskytuje komunikační zabezpečení přes internet a umožňuje aplikacím typu klient/server komunikovat způsobem, který je důvěrný a spolehlivý.

Související informace

Podpora TLS spravovaného klienta IBM MQ.NET

Podpora TLS pro nespravovaného klienta .NET

Podpora TLS pro nespravovaný klient .NET je založena na rozhraní C MQI a sadě GSKit. Rozhraní C MQI zpracovává operace TLS a sada GSKit implementuje protokoly zabezpečeného soketu TLS.

Povolení zabezpečení TLS pro nespravovaného klienta .NET

TLS je podporováno pouze pro připojení klienta. Chcete-li povolit TLS, musíte uvést CipherSpec , která se má použít při komunikaci se správcem front, a to musí odpovídat sadě CipherSpec nastavené na cílovém kanálu.

Chcete-li povolit TLS, uveďte CipherSpec pomocí statické proměnné člena SSLCipherSpec MQEnvironment. Následující příklad se připojuje ke kanálu SVRCONN s názvem SECURE.SVRCONN.CHANNEL, která byla nastavena tak, aby vyžadovala zabezpečení TLS se sadou CipherSpec TLS_RSA_WITH_AES_128_CBC_SHA:

```
MQEnvironment.Hostname         = "your_hostname";
MQEnvironment.Channel          = "SECURE.SVRCONN.CHANNEL";
MQEnvironment.SSLCipherSpec    = "TLS_RSA_WITH_AES_128_CBC_SHA";
MQEnvironment.SSLKeyRepository = "C:\mqm\key";
MQQueueManager qmgr = new MQQueueManager("your_Q_manager");
```

Seznam CipherSpecsnajdete v části Určení CipherSpecs .

Vlastnost SSLCipherSpec může být také nastavena pomocí vlastnosti MQC.SSL_CIPHER_SPEC_PROPERTY v hašovací tabulce vlastností připojení.

Chcete-li se úspěšně připojit pomocí TLS, musí být úložiště klíčů klienta nastaveno s kořenovým řetězcem certifikátů vydavatele certifikátů, ze kterého lze ověřit certifikát prezentovaný správcem front. Podobně, je-li vlastnost SSLClientAuth v kanálu SVRCONN nastavena na hodnotu MQSSL_CLIENT_AUTH_REQUIRED, musí úložiště klíčů klienta obsahovat identifikační osobní certifikát, kterému správce front důvěřuje.

Použití rozlišujícího názvu správce front

Správce front sám identifikuje použití certifikátu TLS, který obsahuje *rozlišující název* (DN).

Aplikace klienta produktu IBM MQ .NET může použít toto DN k ujištění, že komunikuje se správným správcem front. Vzorek DN se zadává pomocí proměnné názvu sslPeerproměnné MQEnvironment. Například nastavení:

```
MQEnvironment.SSLPeerName = "CN=QMGR.*, OU=IBM, OU=WEBSPHERE";
```

umožňuje úspěšné připojení k úspěchu pouze v případě, že správce front předloží certifikát s názvem Common Name začínajícím QMGR., a alespoň dva názvy organizačních jednotek, přičemž první z nich musí být IBM a druhý WEBSPHERE.

Vlastnost SSLPeerName může být také nastavena pomocí vlastnosti MQC.SSL_PEER_NAME_PROPERTY v hašovací tabulce vlastností připojení. Další informace o rozlišujících názvech a pravidlech pro nastavení názvů rovnocenných uzlů naleznete v tématu Zabezpečení IBM MQ.

Je-li parametr SSLPeerName nastaven, úspěšná připojení jsou úspěšná pouze v případě, že je nastavena na platný vzor a správce front představuje odpovídající certifikát.

Ošetření chyb při použití TLS

Při připojování ke správci front s použitím TLS může produkt IBM MQ classes for .NET vydat následující kódy příčiny:

MQRC_SSL_NOT_ALLOWED
Vlastnost SSLCipherSpec byla nastavena, ale bylo použito připojení vazeb. TLS podporuje pouze připojení klienta.

NESROVNALOST MQRC_SSL_PEER_NAME_
Vzorek DN určený ve vlastnosti SSLPeerName neodpovídal rozlišujícímu názvu představenému správcem front.

CHYBA MQRC_SSL_PEER_NAME_ERROR
Vzorek DN zadaný ve vlastnosti SSLPeerName není platný.

Podpora TLS pro spravovaného klienta .NET

Spravovaný klient .NET používá knihovny Microsoft.NET Framework k implementaci protokolů zabezpečených soketů TLS. Třída Microsoft System.Net.SecuritySslStream funguje jako proud přes připojené sokety TCP a odesílá a přijímá data přes toto soketové připojení.

Minimální požadovaná úroveň rámce .NET Framework je .NET Framework v3.5. Úroveň podpory algoritmu šifry je založena na úrovni rámce produktu .NET , kterou aplikace používá.

- Pro aplikace založené na úrovni .NET Framework úrovně 3.5 a v4.0jsou dostupné zabezpečené protokoly soketů SSLv3.0 a TSL v1.0.
- Pro aplikace založené na produktu .NET Framework level4.5jsou dostupné protokoly zabezpečeného soketu: SSLv3.0, TLS v1.1 a TLSv1.2.

Možná budete muset přesunout aplikace, které očekávají vyšší protokol TLS, až po novější verzi rámce, jak je definováno pro podporu zabezpečení produktu Microsoft v produktu .NET Framework.

Hlavní funkce podpory TLS pro spravovaného klienta .NET jsou následující:

Podpora protokolu TLS
Podpora TLS pro spravovaného klienta .NET je definována prostřednictvím třídy .NET SSLStream a závisí na struktuře produktu .NET , kterou aplikace používá. Další informace viz "Podpora protokolu TLS pro spravovaného klienta .NET" na stránce 546.

Podpora CipherSpec
Nastavení TLS pro .NET spravovaného klienta jsou stejná jako u Microsoft.NET TLS steams. Další informace naleznete v tématech "Podpora CipherSpec pro spravovaného klienta .NET" na stránce 546 a "Mapování CipherSpec pro spravovaného klienta .NET" na stránce 547.

Klíčová úložiště
Úložiště klíčů na straně klienta je úložiště klíčů produktu Windows . Úložiště na straně serveru je typem úložiště Cryptographic Message Syntax (CMS). Další informace viz "Klíčová úložiště pro spravovaného klienta .NET" na stránce 549.

Certifikáty
Certifikáty TLS s vlastním podpisem můžete použít k implementaci vzájemného ověření mezi klientem a správcem front. Další informace viz "Použití certifikátů pro spravovaného klienta .NET" na stránce 549.

SSLPEERNAME
V produktu .NETmohou aplikace používat volitelný atribut SSLPEERNAME k určení vzorku rozlišovacího jména (DN). Další informace viz "SSLPEERNAME" na stránce 550.

shoda s normou FIPS
Povolení standardu FIPS není podporováno knihovnou zabezpečení produktu Microsoft.NET . Povolení FIPS je řízeno nastavením zásad skupiny Windows .

Shoda NSA Suite B
IBM MQ implementuje RFC 6460. Implementace produktu Microsoft.NET pro sadu NSA Suite B je 5430. Tento stav je podporován z rámce .NET Framework 3.5 .

Obnovení tajného klíče nebo opětovné vyjednávání
Přestože třída SSLStream nepodporuje opětovné nastavení tajného klíče nebo opětovné vyjednávání, pro konzistenci s ostatními klienty IBM MQ umožňuje spravovaný klient .NET nastavit počet SSLKeyReset. Další informace viz "Obnovení tajného klíče nebo opětovné vyjednávání" na stránce 550.

Kontrola odvolání
Třída SSLStream podporuje kontrolu odvolání certifikátů, která se automaticky provádí pomocí stroje pro získávání certifikátů. Další informace viz "Kontrola odvolání" na stránce 550.

Podpora uživatelské procedury zabezpečení produktu IBM MQ

Třída SSLStream poskytuje omezenou podporu pro ukončení zabezpečení produktu IBM MQ . Dotazování na lokální a vzdálené certifikáty pro získání SSLPeerNamePtr (Subject DN) a SSLRemCertIssNamePtr (Issuer DN) je možné, protože toto je podporováno v Microsoft.NET. Avšak neexistuje žádná podpora pro získání atributů, jako je DNQ, UNSTRUCTURE_DNAME a UNSTRUCTUREODADDRESS, takže tyto hodnoty nelze načíst pomocí ukončení.

Podpora kryptografického hardwaru

Šifrovací hardware není pro spravovaného klienta .NET podporován.

Podpora protokolu TLS pro spravovaného klienta .NET

Podpora TLS IBM MQ.NET je založena na třídě .NET SSLStream.

Poznámka: Podpora protokolu TLS pro spravovaného klienta .NET závisí na úrovni rámce produktu .NET , kterou aplikace používá. Další informace viz "Podpora TLS pro spravovaného klienta .NET" na stránce 545.

Aby byla třída SSLStream produktu Microsoft.NET inicializována TLS a provedla ruční navázání spojení se správcem front, jeden z povinných parametrů, které musíte nastavit, je **SSLProtocol**, kde musíte uvést číslo verze TLS, které musí být jedna z následujících hodnot:

- SSL3.0
- TLS1.0
- TLS1.2

Hodnota tohoto parametru je úzce svázána s rodinou protokolu, do které patří upřednostňovaná položka CipherSpec . Když se SSLStream spustí navázání komunikace TLS se serverem (správce front), použije verzi TLS uvedenou v produktu **SSLProtocol** k identifikaci seznamu CipherSpecs , které mají být použity pro dohadování.

Produkt IBM MQ.NET nečiní žádné vlastnosti, které jsou k dispozici pro aplikace určené k nastavení této hodnoty. Místo toho IBM MQ používá mapovací tabulku k interní mapování sady CipherSpec nastavené na řadu protokolů a identifikuje verzi protokolu SSLProtocol, která má být použita. Tato tabulka obsahuje mapování každé z podporovaných CipherSpec mezi Microsoft.NET a IBM MQa verzí protokolu, do které patří. Další informace viz "Mapování CipherSpec pro spravovaného klienta .NET" na stránce 547.

Podpora CipherSpec pro spravovaného klienta .NET

Nastavení CipherSpec pro aplikaci se používá během navázání komunikace se serverem.

Klienti produktu IBM MQ umožňují nastavit hodnotu CipherSpec použitou při navázání komunikace se správcem front. Klienti produktu IBM MQ by měli nastavit platnou položku CipherSpec pro zabezpečené připojení, nejlépe CipherSpec uvedenou ve skupině zásad produktu Windows . Ponecháte-li toto pole prázdné, bude kanál bez zabezpečení na soketech označován jako prostý textový kanál.

Pro spravovaného klienta produktu IBM MQ.NET jsou nastavení TLS určena pro třídu Microsoft.NET SSLStream. Pro SSLStream, CipherSpecnebo seznam předvoleb CipherSpecslze nastavit pouze v zásadě skupiny Windows , která je celopočítačovým nastavením. SSLStream pak použije zadaný seznam CipherSpec nebo seznam předvoleb během navázání komunikace se serverem. V případě jiných klientů IBM MQ lze vlastnost CipherSpec nastavit v aplikaci v definici kanálu produktu IBM MQ a stejné nastavení se používá pro vyjednávání TLS. V důsledku tohoto omezení může komunikace výměnou potvrzení TLS vyjednávat jakoukoli podporovanou specifikaci CipherSpec bez ohledu na to, co je uvedeno v konfiguraci kanálu produktu IBM MQ . Proto je pravděpodobné, že to bude mít za následek chybu AMQ9631 ve správci front. Chcete-li se této chybě vyhnout, nastavte stejnou hodnotu CipherSpec jako ta, kterou jste nastavili v aplikaci jako konfiguraci TLS v zásadě skupiny produktu Windows .

The new IBM MQ.NET TLS client code checks only that the correct protocol version was negotiated. Verze protokolu TLS je odvozena ze sady CipherSpec , kterou aplikace nastavuje a používá se pro navázání komunikace TLS se serverem (správcem front). Proto je vyžadována návrhem na nastavení hodnoty CipherSpec v aplikaci klienta spravovaného produktem IBM MQ.NET . Je-li sada CipherSpec nastavená klientem produktu IBM MQ jiná než verze z protokolů SSL 3.0, TLS 1.0 a TLS 1.2 , klient IBM MQ spravovaný .NET bude standardně vyjednávat se všemi šifry z protokolů SSL3.0 nebo TLS1.0 a nenahlásí chybu.

Poznámka: Pokud hodnota CipherSpec dodaná aplikací není známá jako CipherSpec IBM MQ, spravovaný .NET klient IBM MQ jej nevnímá a vyjedná připojení založené na zásadě skupiny systému Windows .

Nastavení CipherSpec

Existují tři způsoby, jak nastavit CipherSpec:

Třída MQEnvironment .NET
Následující příklad ukazuje, jak nastavit CipherSpec se třídou MQEnvironment.

```
MQEnvironment.SSLKeyRepository = "*USER";
MQEnvironment.ConnectionName = connectionName;
MQEnvironment.Channel = channelName;
MQEnvironment.properties.Add(MQC.TRANSPORT_PROPERTY, MQC.TRANSPORT_MQSERIES_MANAGED);
MQEnvironment.SSLCipherSpec = "TLS_RSA_WITH_AES_128_CBC_SHA";
```

Vlastnost TLS CipherSpec
Následující příklad ukazuje, jak nastavit parametr CipherSpec přidáním parametru hashtable do konstruktoru MQQueueManager .

```
properties = new Hashtable();
properties.Add(MQC.TRANSPORT_PROPERTY, MQC.TRANSPORT_MQSERIES_MANAGED);
properties.Add(MQC.HOST_NAME_PROPERTY, hostName);
properties.Add(MQC.PORT_PROPERTY, port);
properties.Add(MQC.CHANNEL_PROPERTY, channelName);
properties.Add(MQC.SSL_CERT_STORE_PROPERTY, sslKeyRepository);
properties.Add(MQC.SSL_CIPHER_SPEC_PROPERTY, cipherSpec);
properties.Add(MQC.SSL_PEER_NAME_PROPERTY, sslPeerName);
properties.Add(MQC.SSL_RESET_COUNT_PROPERTY, keyResetCount);
queueManager = new MQQueueManager(queueManagerName, properties);
```

Zásada skupiny Windows
Je-li sada CipherSpec nastavena na zásadu skupiny Windows , musí být nastavena stejná položka CipherSpec pro hodnotu vlastnosti SSLCipherSpec v kanálu SVRCONN a v aplikaci. Je-li zásada skupiny Windows nastavena na výchozí hodnotu, tj. zásada skupiny není povolena/upravována pro nastavení CipherSpec , aplikace musí nastavit stejnou výchozí hodnotu CipherSpec z konfigurace TLS zásad skupiny produktu Windows ve třídě MQEnvironment nebo ve vlastnostech hašovací tabulky konstruktoru MQQueueManager .

Použití CCDT

Produkt IBM MQ.NET podporuje pouze tabulky definic kanálů klienta (soubory .TAB), které se nacházejí na lokálním počítači. Existující soubory CCDT, které mají sadu hodnot CipherSpec , lze použít pro připojení IBM MQ.NET . Hodnota protokolu CipherSpec na kanálu připojení klienta však určuje verzi protokolu TLS a také musí odpovídat sadě CipherSpec nastavenou ve skupině zásad produktu Windows .

Související pojmy
“Nastavení prostředí produktu IBM MQ” na stránce 535
Než použijete připojení klienta k připojení ke správci front, je třeba nastavit prostředí produktu IBM MQ .

Související informace
Určení specifikace CipherSpecs
Třída MQEnvironment .NET

Mapování CipherSpec pro spravovaného klienta .NET
Rozhraní IBM MQ.NET udržuje mapovací tabulku IBM MQ na Microsoft.NET , která se používá k určení verze protokolu TLS, kterou musí spravovaný klient použít k vytvoření zabezpečeného připojení se správcem front.

Je-li v kanálu SVRCONN zadána hodnota CipherSpec , pokusí se správce front po dokončení navázání komunikace TLS porovnat tuto položku CipherSpec s vyjednanou CipherSpec , kterou používá klientská

aplikace. Pokud správce front nemůže najít odpovídající CipherSpec, komunikace selže s chybou AMQ9631.

Rozhraní IBM MQ.NET spravuje mapovací tabulku IBM MQ až Microsoft.NET CipherSpec . Tato tabulka se používá k určení verze protokolu TLS, kterou chce klient použít k vytvoření zabezpečeného soketového připojení ke správci front. Na základě hodnoty SSLCipherSpec může být verze protokolu SSLProtocol TLS v1.0nebo TLS v1.2, v závislosti na tom, kterou verzi produktu Microsoft.NET Framework používáte.

Ujistěte se, že jste poskytli správnou hodnotu SSLCipherSpec jako určení nesprávné hodnoty, což může vést k použití protokolů SSL3.0 nebo TLS1.0 .

Tabulka 76. Tabulka mapování IBM MQ a Microsoft.NET

IBM MQ CipherSpec	**Microsoft.NET CipherSpec**	**Verze TLS**
TLS_RSA_WITH_AES_128_CBC_SHA	TLS_RSA_WITH_AES_128_CBC_SHA	TLS 1.0
TLS_RSA_WITH_AES_256_CBC_SHA	TLS_RSA_WITH_AES_256_CBC_SHA	TLS 1.0
TLS_RSA_WITH_3DES_EDE_CBC_SHA [1]	TLS_RSA_WITH_3DES_EDE_CBC_SHA [1]	TLS 1.0
TLS_RSA_WITH_AES_128_CBC_SHA256	TLS_RSA_WITH_AES_128_CBC_SHA256	TLS 1.2
TLS_RSA_WITH_AES_256_CBC_SHA256	TLS_RSA_WITH_AES_256_CBC_SHA256	TLS 1.2
ECDHE_RSA_AES_128_CBC_SHA256	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256_P256	TLS 1.2
ECDHE_RSA_AES_128_CBC_SHA256	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256_P384	TLS 1.2
ECDHE_RSA_AES_128_CBC_SHA256	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256_P521	TLS 1.2
ECDHE_ECDSA_AES_128_CBC_SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256_P256	TLS 1.2
ECDHE_ECDSA_AES_128_CBC_SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256_P384	TLS 1.2
ECDHE_ECDSA_AES_128_CBC_SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256_P521	TLS 1.2
ECDHE_ECDSA_AES_256_CBC_SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384_P384	TLS 1.2
ECDHE_ECDSA_AES_256_CBC_SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384_P521	TLS 1.2
ECDHE_ECDSA_AES_128_GCM_SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256_P256	TLS 1.2
ECDHE_ECDSA_AES_128_GCM_SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256_P384	TLS 1.2
ECDHE_ECDSA_AES_128_GCM_SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256_P521	TLS 1.2
ECDHE_ECDSA_AES_256_GCM_SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384_P384	TLS 1.2
ECDHE_ECDSA_AES_256_GCM_SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384_P521	TLS 1.2

Notes:

1. Tato CipherSpec TLS_RSA_WITH_3DES_EDE_CBC_SHA byla zamítnuta. Nicméně lze ji přesto použít k přenosu až 32 GB dat před ukončením připojení s chybou AMQ9288. Chcete-li se této chybě vyhnout, je třeba při použití této CipherSpecbuď zabránit použití trojitého DES, nebo povolit resetování tajného klíče.

Klíčová úložiště pro spravovaného klienta .NET
Úložiště klíčů použité spravovanými klienty .NET je úložiště klíčů produktu Windows . Certifikáty a soukromé klíče musí být dostupné buď v uživatelské nebo systémové úložišti klíčů, aby mohl být používán klientskou aplikací pro identitu a důvěryhodnost během komunikace výměnou potvrzení TLS.

Strana klienta

V aplikaci můžete nastavit jednu z následujících hodnot pro úložiště klíčů:

- "*USER": Produkt IBM MQ.NET přistupuje k úložišti certifikátů aktuálního uživatele, aby bylo možné načíst certifikáty klienta.
- "*SYSTEM": Produkt IBM MQ.NET přistupuje k účtu lokálního počítače za účelem načtení certifikátů.

Certifikáty klienta musí být uloženy v úložišti Moje paměť uživatele nebo počítačového účtu. Všechny certifikáty serveru (CA) musí být uloženy v kořenovém adresáři úložiště certifikátů.

Poznámka: Do jednoho souboru můžete uložit více než jeden certifikát v následujících formátech:

- Personal Information Exchange-PKCS #12 (.PFX, .P12)
- Syntax Message Syntax Standard-PKCS #7 Certifikáty (.P7B)
- Microsoft Serializované úložiště certifikátů (.SST)

Použití certifikátů pro spravovaného klienta .NET
For client certificates, the IBM MQ managed .NET client accesses the Windows keystore and loads all of the client's certificates that are matched either by certificate label or matched by the string.

Při výběru certifikátu, který má být použit, klient IBM MQ spravovaný .NET vždy použije první vyhovující certifikát pro navázání komunikace TLS SSL.

Vyhovující certifikáty podle štítku certifikátu

Nastavíte-li jmenovku certifikátu, klient IBM MQ spravovaných .NET prohledá úložiště certifikátů Windows s daným názvem štítku, aby identifikoval certifikát klienta. Načte všechny odpovídající certifikáty a použije první certifikát na seznamu. Pro nastavení štítku certifikátu jsou k dispozici dvě volby:

- Popisek certifikátu může být nastaven na třídu MQEnvironment s přístupem k objektu MQEnvironment.CertificateLabel.
- Popisek certifikátu může být také nastaven ve vlastnostech hašovací tabulky, který je dodáván jako vstupní parametr konstruktoru MQQueueManager , jak je uvedeno v následujícím příkladu.

```
Hashtable properties = new Hashtable();
properties.Add("CertificateLabel", "mycert");
```

Název ("CertificateLabel") a hodnota je citlivá na velikost písmen.

Odpovídající certifikáty podle řetězce

Není-li popisek certifikátu nastaven, vyhledá se a použije certifikát, který odpovídá řetězci `"ibmwebspheremq"` a aktuálně přihlášenému uživateli (ve malých písmenech).

Související informace
Třída MQEnvironment .NET
Bezpečná připojení klienta ke správci front

SSLPEERNAME
Atribut SSLPEERNAME se používá ke kontrole rozlišujícího názvu (Distinguished Name-DN) certifikátu od správce front typu peer.

V produktu IBM MQ.NETmohou aplikace používat parametr SSLPEERNAME k určení vzoru rozlišujícího názvu, jak je uvedeno v následujícím příkladu.

```
SSLPEERNAME(CN=QMGR.*, OU=IBM, OU=WEBSPHERE)
```

Podobně jako u jiných klientů produktu IBM MQ je parametr SSLPEERNAME volitelným parametrem.

Není-li hodnota SSLPEERNAME nastavena, spravovaný klient IBM MQ.NET neprovede žádné ověření platnosti certifikátu vzdáleného serveru (Server) a spravovaný klient pouze přijme jako-je certifikát Remote (/server).

Způsob, jakým nastavíte parametr SSLPEERNAME, závisí na tom, kterou z nabídek zásobníku produktu IBM MQ používáte.

IBM MQ classes for .NET
Existují tři možnosti, jak je uvedeno níže.

1. Nastavte položku MQEnvironment.SSLPeerName ve třídě MQEnvironment.
2. `MQEnvironment.properties.Add(MQC.SSL_PEER_NAME_PROPERTY, value)`
3. Použijte konstruktor správce front `MQQueueManager (String queueManagerName, Hashtable properties)`. Zadejte parametr SSLPEERNAME v souboru `Hashtable properties` jako volbu 2.

XMS .NET
Nastavte název partnera SSL v továrně připojení:

```
ConnectionFactory.SetStringProperty(XMSC.WMQ_SSL_PEER_NAME, value);
```

WCF
Do pole URI zahrňte název SslPeerjako středníky oddělené pole.

Související informace
Třída MQEnvironment .NET

Obnovení tajného klíče nebo opětovné vyjednávání
Třída SSLStream nepodporuje opětovné sečtení/opětovné vyjednávání tajného klíče. However, to be consistent with other IBM MQ clients, the IBM MQ managed .NET client allows applications to set SSLKeyResetCount.

Když je dosaženo limitu, produkt IBM MQ.NET se odpojí od správce front a aplikace obdrží oznámení o této výjimce jako o výjimce MQRC_CONNECTION_BROKEN jako kód příčiny. Aplikace mohou pracovat s výjimkou a znovu zavést připojení nebo povolit volbu MQCNO_RECONNECT pro produkt IBM MQ.NET pro automatické opětovné připojení ke správci front.

Povolení prostředku automatického opětného připojení klienta znamená, že je-li dosažen počet resetování klíče, všechna existující připojení jsou ukončena a klient IBM MQ.NET znovu vytvoří všechna připojení znovu. Další informace o automatickém opětovném připojení klienta najdete v tématu Automatické opětovné připojení klienta.

Související informace
Resetování tajných klíčů TLS

Kontrola odvolání
Třída SSLStream podporuje kontrolu odvolání certifikátů.

Kontrola odvolání je automaticky prováděna pomocí stroje pro řetězení certifikátů. To platí jak pro protokol OCSP (Online Certificate Status Protocol), tak pro seznamy odvolaných certifikátů (CRL). Třída SSLStream používá odvolání certifikátů, které používá pouze server uvedený v certifikátu, tj. server je

diktován vlastním certifikátem. Je možné, aby rozšíření CDP HTTP a HTTP požadavky HTTP proxy byly proxy prostřednictvím serveru proxy HTTP.

Způsob, jakým nastavíte kontrolu odvolání, závisí na tom, které z nabídek zásobníku produktu IBM MQ používáte.

IBM MQ.NET
Kontrola odvolání může být nastavena přístupem k vlastnosti **MQEnvironment.SSLCertRevocationCheck** v souboru třídy `MQEnvironment.cs` .

XMS .NET
Kontrola odvolání může být nastavena na kontext vlastnosti továrny připojení, jak je uvedeno v následujícím příkladu.

```
ConnectionFactory.SetBooleanProperty(XMSC.WMQ_SSL_CERT_REVOCATION_CHECK, true);
```

WCF
Kontrola odvolání může být nastavena na identifikátoru URI pomocí následující konvence pojmenování.

```
"SslCertRevocationCheck=true"
```

Konfigurace TLS pro spravované IBM MQ .NET
Konfigurace protokolu TLS pro spravované IBM MQ .NET spočívá ve vytvoření certifikátů podepsaného, pak konfiguraci strany serveru, straně klienta a aplikačního programu.

Informace o této úloze

Chcete-li nakonfigurovat TLS, musíte nejprve vytvořit příslušné certifikáty podepsaného. Certifikáty podepisujících subjektů mohou být buď samy podepsané, nebo certifikáty poskytnuté certifikační autoritou. Ačkoli certifikáty podepsané sebou samým lze použít na vývojovém, testovacím nebo předprovozním systému, nepoužívejte je na produkčním systému. V provozním systému použijte certifikáty, které jste získali od důvěryhodné externí certifikační autority (CA).

Postup

1. Vytvořte certifikáty podepsaného.
   a) Chcete-li vytvořit certifikáty podepsané sebou samým, použijte jeden z následujících nástrojů poskytnutých s produktem IBM MQ :

      Použijte buď grafické uživatelské rozhraní produktu **strmqikm** , nebo **runmqckm** nebo **runmqakm** z příkazového řádku. Další informace o použití těchto nástrojů najdete v tématu Použití produktů **runmqckm**, **runmqakm**a **strmqikm** ke správě digitálních certifikátů.
   b) Chcete-li získat certifikáty pro správce front a klienty z certifikační autority (CA), postupujte podle pokynů v tématu Získání osobních certifikátů od certifikační autority.
2. Nakonfigurujte stranu serveru.
   a) Nakonfigurujte TLS ve správci front pomocí sady GSKit, jak je popsáno v tématu Bezpečně připojit klienta ke správci front.
   b) Nastavte atributy protokolu TLS kanálu SVRCONN:
      - Nastavte **SSLCAUTH** na `"REQUIRED/OPTIONAL"`.
      - Nastavte **SSLCIPH** na odpovídající CipherSpec.

      Další informace viz “Povolení zabezpečení TLS pro nespravovaného klienta .NET” na stránce 544.
3. Nakonfigurujte stranu klienta.
   a) Importujte certifikáty klienta do úložiště certifikátů produktu Windows (pod účtem Uživatel/Počítač).

Produkt IBM MQ .NET přistupuje k certifikátům klienta z úložiště certifikátů produktu Windows , proto je třeba importovat certifikáty do úložiště certifikátů produktu Windows pro vytvoření zabezpečeného soketového připojení k produktu IBM MQ . Další informace o tom, jak přistupovat k úložišti klíčů produktu Windows a importovat certifikáty na straně klienta, najdete v tématu Import nebo export certifikátů a soukromých klíčů.

b) Zadejte vlastnost CertificateLabel , jak je popsáno v tématu Bezpečný připojení klienta ke správci front.

c) V případě potřeby upravte zásadu skupiny produktu Windows a nastavte ji CipherSpec, poté, aby se aktualizace zásad skupiny produktu Windows projevily, restartujte počítač.

4. Konfigurujte aplikační program.

a) Nastavte hodnotu připojení MQEnvironment nebo SSLCipherSpec tak, aby jako zabezpečené připojení bylo označeno připojení.

Hodnota, kterou zadáte, se použije pro identifikaci protokolu, který se používá (TLS). Sada CipherSpec by měla být jednou z CipherSpecs podporované verze protokolu SSLProtocol a může být pokud možno stejná jako ta, která je uvedena v zásadě skupiny produktu Windows . (Podporovaná verze protokolu SSLProtocol závisí na použitém rámci .NET . Verze protokolu SSLProtocol může být TLS v1.0, nebo TLS v1.2, v závislosti na tom, kterou verzi produktu Microsoft .NET Framework používáte.)

Poznámka: Pokud hodnota CipherSpec dodaná aplikací není známá jako CipherSpec IBM MQ, spravovaný .NET klient IBM MQ jej nevnímá a vyjedná připojení založené na zásadě skupiny systému Windows .

b) Nastavte vlastnost SSLKeyRepository na hodnotu "*SYSTEM" nebo "*USER".

c) Volitelné: Nastavte parametr SSLPEERNAME na rozlišující název (DN) certifikátu serveru.

d) Zadejte vlastnost CertificateLabel , jak je popsáno v tématu Bezpečný připojení klienta ke správci front.

e) Nastavte jakékoli další volitelné parametry, které vyžadujete jako KeyResetCount, CertificationRevocationCheck, a povolte standard FIPS.

Příklady nastavení způsobu nastavení protokolu TLS a úložiště klíčů TLS

Pro produkt Base .NETmůžete nastavit protokol TLS a úložiště klíčů TLS prostřednictvím třídy MQEnvironment, jak je uvedeno v následujícím příkladu:

```
MQEnvironment.SSLCipherSpec = "TLS_RSA_WITH_AES_128_CBC_SHA256";
MQEnvironment.SSLKeyRepository = "*USER";

MQEnvironment.properties.Add(MQC.SSL_CIPHER_SPEC_PROPERTY, "TLS_RSA_WITH_AES_128_CBC_SHA256")
```

Případně můžete nastavit úložiště klíčů TLS a TLS zadáním hašovací tabulky jako součásti konstruktoru MQQueueManager , jak je uvedeno v následujícím příkladu.

```
Hashtable properties = new Hashtable();
properties.Add(MQC.SSL_CERT_STORE_PROPERTY, sslKeyRepository);
properties.Add(MQC.SSL_CIPHER_SPEC_PROPERTY, "TLS_RSA_WITH_AES_128_CBC_SHA256")
```

Jak pokračovat dále

Další informace o tom, jak začít pracovat s vývojem aplikací správy TLS IBM MQ .NET , viz“Psaní jednoduché aplikace” na stránce 553.

Související informace

Třída MQEnvironment .NET

Počet KeyReset(MQLONG)

Federální standardy zpracování informací (FIPS) pro UNIX, Linux, and Windows

Psaní jednoduché aplikace
Tipy pro zápis jednoduché aplikace .NET TSL IBM MQ , včetně příkladů nastavení vlastností zabezpečení SSL pro továrny připojení, vytvoření instance správce front, připojení, relace a místa určení a odeslání testovací zprávy.

Než začnete

Nejprve musíte nakonfigurovat TLS pro spravovanou IBM MQ.NET , jak je popsáno v “Konfigurace TLS pro spravované IBM MQ .NET” na stránce 551.

Pro konfiguraci aplikačního programu v základním produktu .NETnastavte vlastnosti SSL buď pomocí třídy MQEnvironment, nebo zadáním hašovací tabulky jako části konstruktoru MQQueueManager .

Pro konfiguraci aplikačního programu v produktu XMS .NETnastavte vlastnosti zabezpečení SSL v kontextu vlastnosti továren připojení.

Postup

1. Nastavte vlastnosti zabezpečení SSL pro továrny připojení, jak je uvedeno v následujících příkladech.

   Příklad pro IBM MQ.NET

   ```
   properties = new Hashtable();
   properties.Add(MQC.TRANSPORT_PROPERTY, MQC.TRANSPORT_MQSERIES_MANAGED);
   properties.Add(MQC.HOST_NAME_PROPERTY, hostName);
   properties.Add(MQC.PORT_PROPERTY, port);
   properties.Add(MQC.CHANNEL_PROPERTY, channelName);
   properties.Add(MQC.SSL_CERT_STORE_PROPERTY, sslKeyRepository);
   properties.Add(MQC.SSL_CIPHER_SPEC_PROPERTY, cipherSpec);
   properties.Add(MQC.SSL_PEER_NAME_PROPERTY, sslPeerName);
   properties.Add(MQC.SSL_RESET_COUNT_PROPERTY, keyResetCount);
   properties.Add("CertificateLabel", "ibmwebspheremq");
   MQEnvironment.SSLCertRevocationCheck = sslCertRevocationCheck;
   ```

   Příklad pro XMS .NET

   ```
   cf.SetStringProperty(XMSC.WMQ_SSL_KEY_REPOSITORY, "sslKeyRepository");
   cf.SetStringProperty(XMSC.WMQ_SSL_CIPHER_SPEC, cipherSpec);
   cf.SetStringProperty(XMSC.WMQ_SSL_PEER_NAME, sslPeerName);
   cf.SetIntProperty(XMSC.WMQ_SSL_KEY_RESETCOUNT, keyResetCount);
   cf.SetBooleanProperty(XMSC.WMQ_SSL_CERT_REVOCATION_CHECK, true);
   ```

2. Vytvořte instanci správce front, připojení, relaci a cíl, jak je uvedeno v následujících příkladech.

   Příklad pro produkt MQ .NET

   ```
   queueManager = new MQQueueManager(queueManagerName, properties);
   Console.WriteLine("done");

   // accessing queue
   Console.Write("Accessing queue " + queueName + ".. ");
   queue = queueManager.AccessQueue(queueName, MQC.MQOO_OUTPUT +
   MQC.MQOO_FAIL_IF_QUIESCING);
   Console.WriteLine("done");
   ```

   Příklad pro XMS .NET

   ```
   connectionWMQ = cf.CreateConnection();
   // Create session
   sessionWMQ = connectionWMQ.CreateSession(false, AcknowledgeMode.AutoAcknowledge);

   // Create destination
   destination = sessionWMQ.CreateQueue(destinationName);

   // Create producer
   producer = sessionWMQ.CreateProducer(destination);
   ```

3. Odešlete zprávu, jak je zobrazeno v následujících příkladech.

Příklad pro produkt MQ .NET

```
// creating a message object
message = new MQMessage();
message.WriteString(messageString);

// putting messages continuously
for (int i = 1; i <= numberOfMsgs; i++)
{
Console.Write("Message " + i + " <" + messageString + ">.. ");
queue.Put(message);
Console.WriteLine("put");
}
```

Příklad pro XMS .NET

```
textMessage = sessionWMQ.CreateTextMessage();
textMessage.Text = simpleMessage;
producer.Send(textMessage);
```

4. Ověřte připojení TLS.

   Zkontrolujte stav kanálu a ověřte, zda bylo navázáno spojení TLS a zda funguje správně.

Konfigurace trasování pro SSLStream
Chcete-li zachytit události trasování a zprávy týkající se třídy SSLStream, musíte přidat sekci konfigurace pro diagnostiku systému do konfiguračního souboru aplikace pro vaši aplikaci.

Informace o této úloze

Nepřidáte-li sekci konfigurace pro diagnostiku systému do konfiguračního souboru aplikace, klient IBM MQ spravovaný .NET nebude zachycovat žádné události, trasování nebo body ladění týkající se TLS a třídy SSLStream.

Poznámka: Spuštění trasování produktu IBM MQ pomocí produktu **strmqtrc** nezachytí všechny požadované trasování TLS.

Postup

1. Vytvořte konfiguraci aplikace (App.Config) pro projekt aplikace.
2. Přidejte sekci konfigurace diagnostiky systému, jak je uvedeno v následujícím příkladu.

```
<system.diagnostics>
           <sources>
                 <source name="System.Net" tracemode="includehex">
                       <listeners>
                             <add name="ExternalSourceTrace"/>
                       </listeners>
                 </source>
                 <source name="System.Net.Sockets">
                       <listeners>
                             <add name="ExternalSourceTrace"/>
                       </listeners>
                 </source>
                 <source name="System.Net.Cache">
                       <listeners>
                             <add name="ExternalSourceTrace"/>
                       </listeners>
                 </source>
                 <source name="System.Net.Security">
                       <listeners>
                             <add name="ExternalSourceTrace"/>
                       </listeners>
                 </source>
                 <source name="System.Security">
                       <listeners>
                             <add name="ExternalSourceTrace"/>
                       </listeners>
                 </source>
           </sources>
```

```
            <switches>
                  <add name="System.Net" value="Verbose"/>
                  <add name="System.Net.Sockets" value="Verbose"/>
                  <add name="System.Net.Cache" value="Verbose"/>
                  <add name="System.Security" value="Verbose"/>
                  <add name="System.Net.Security" value="Verbose"/>
            </switches>

            <sharedListeners>
                  <add name="ExternalSourceTrace" type="IBM.WMQ.ExternalSourceTrace,
amqmdnet, Version=n.n.n.n, Culture=neutral, PublicKeyToken=dd3cb1c9aae9ec97"/>
            </sharedListeners>
         <trace autoflush="true"/>
     </system.diagnostics>
```

Upozornění: Pole `Version` položky add name musí být kterákoli verze souboru `amqmdnet.dll` .net, která se používá.

Ukázkové aplikace pro implementaci TLS ve spravovaném .NET
K dispozici jsou ukázkové aplikace pro zobrazení implementace TLS pro spravovanou .NET v IBM MQ classes for .NET, XMS .NET a IBM MQ vlastního kanálu pro WCF.

V následující tabulce jsou uvedeny umístění ukázkových aplikací. *MQ_INSTALLATION_PATH* představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Tabulka 77. Umístění ukázkových aplikací pro implementaci TLS ve spravovaném .NET

Nabídka zásobníku produktu IBM MQ.NET	**Umístění vzorků**
základní.NET	*MQ_INSTALLATION_PATH*\Tools\dotnet\samples\cs\base\SimplePut\SimplePut.cs *MQ_INSTALLATION_PATH*\Tools\dotnet\samples\cs\base\SimpleGet\SimpleGet.cs
XMS .NET	*MQ_INSTALLATION_PATH*\Tools\dotnet\samples\cs\xms\simple\wmq\SimpleProducer\SimpleProducer.cs *MQ_INSTALLATION_PATH*\Tools\dotnet\samples\cs\xms\simple\wmq\SimpleConsumer\SimpleConsumer.cs
Vlastní kanál produktu IBM MQ pro prostředek WCF	*MQ_INSTALLATION_PATH*\Tools\dotnet\samples\cs\wcf\samples\WCF\oneway\service\MQMessagingOneWayService.cs

Windows Použití monitoru .NET

Monitor .NET je aplikace podobná monitoru spouštěčů IBM MQ .

Důležité: See Features that can be used only with the primary installation on Windows for important information.

Můžete vytvořit komponenty produktu .NET , které jsou převedeny na instanci, kdykoli je zpráva přijata na monitorované frontě a která pak tuto zprávu zpracuje. Monitor .NET se spustí příkazem **runmqdnm** a zastaví se příkazem **endmqdnm** . Podrobné informace o těchto příkazech naleznete v příručce runmqdnm a endmqdnm.

Chcete-li použít produkt .NET Monitor, napiš komponentu, která implementuje rozhraní IMQObjectTrigger , která je definována v souboru amqmdnm.dll.

Komponenty mohou být buď transakční, nebo netransakční. Transakční komponenta musí být zděděna z System.EnterpriseServices.ServicedComponent a musí být registrována buď jako RequiresTransaction , nebo SupportsTransaction. Nesmí být registrována jako RequiresNew , protože monitor .NET již zahájil transakci.

Komponenta přijímá objekty MQQueueManager, MQQueue a MQMessage z produktu **runmqdnm**. Může také přijmout řetězec s parametry uživatele, pokud byl zadán, pomocí volby příkazového řádku *-u* , když bylo spuštěno runmqdnm . Všimněte si, že vaše komponenta přijímá obsah zprávy, která byla doručena do monitorované fronty v objektu MQMessage. Nemusí se připojovat ke správci front, otevřít frontu nebo získat zprávu samotnou. Komponenta pak musí zpracovat zprávu jako odpovídající a vrátit řízení do monitoru .NET .

Pokud byla vaše komponenta zapsána jako transakční komponenta, zaregistruje se k potvrzení nebo odvolání transakce pomocí funkcí poskytovaných produktem System.EnterpriseServices.ServicedComponent.

Protože komponenta přijímá objekty MQQueueManager a MQQueue, stejně jako zprávu, má pro tuto zprávu úplné informace o kontextu a může například otevřít jinou frontu ve stejném správci front, aniž by bylo nutné se samostatně připojit k produktu IBM MQ.

Windows *Příklad fragmentů kódu*

Toto téma obsahuje dva příklady komponent, které obdrží zprávu z produktu .NET Monitor a vytisknou ji, jednu pomocí transakčního zpracování a ostatního netransakčního zpracování. Třetí příklad ukazuje běžné rutiny obslužných programů, které jsou použitelné pro první dva příklady. Všechny příklady jsou v C#.

Příklad 1: Transakční zpracování

```
/********************************************************************/
/* Licensed materials, property of IBM                              */
/* 63H9336                                                          */
/* (C) Copyright IBM Corp. 2005, 2023.                              */
/********************************************************************/
using System;
using System.EnterpriseServices;

using IBM.WMQ;
using IBM.WMQMonitor;

[assembly: ApplicationName("dnmsamp")]

// build:
//
// csc -target:library -reference:amqmdnet.dll;amqmdnm.dll TranAssembly.cs
//
// run (with dotnet monitor)
//
// runmqdnm -m QMNAME -q QNAME -a dnmsamp.dll -c Tran

namespace dnmsamp
{
  [TransactionAttribute(TransactionOption.Required)]
  public class Tran : ServicedComponent, IMQObjectTrigger
  {
    Util util = null;

    [AutoComplete(true)]
    public void Execute(MQQueueManager qmgr, MQQueue queue,
        MQMessage message, string param)
    {
      util = new Util("Tran");

      if (param != null)
        util.Print("PARAM: '" +param.ToString() + "'");

      util.PrintMessage(message);

      //System.Console.WriteLine("SETTING ABORT");
      //ContextUtil.MyTransactionVote = TransactionVote.Abort;

      System.Console.WriteLine("SETTING COMMIT");
      ContextUtil.SetComplete();
      //ContextUtil.MyTransactionVote = TransactionVote.Commit;
    }
  }
}
```

Příklad 2: Netransakční zpracování

```
/*********************************************************************/
/* Licensed materials, property of IBM                               */
/* 63H9336                                                           */
/* (C) Copyright IBM Corp. 2005, 2023.                               */
/*********************************************************************/

using System;

using IBM.WMQ;
using IBM.WMQMonitor;

// build:
//
// csc -target:library -reference:amqmdnet.dll;amqmdnm.dll NonTranAssembly.cs
//
// run (with dotnet monitor)
//
// runmqdnm -m QMNAME -q QNAME -a dnmsamp.dll -c NonTran
namespace dnmsamp
{
  public class NonTran : IMQObjectTrigger
  {
    Util util = null;

    public void Execute(MQQueueManager qmgr, MQQueue queue,
        MQMessage message, string param)
    {
      util = new Util("NonTran");

      try
      {
        util.PrintMessage(message);
      }

      catch (Exception ex)
      {
        System.Console.WriteLine(">>> NonTran\n{0}", ex.ToString());
      }
    }
  }
}
```

Příklad 3: Běžné rutiny

```
/*********************************************************************/
/* Licensed materials, property of IBM                               */
/* 63H9336                                                           */
/* (C) Copyright IBM Corp. 2005, 2023.                               */
/*********************************************************************/

using System;

using IBM.WMQ;

namespace dnmsamp
{
    /// <summary>
    /// Summary description for Util.
    /// </summary>
    public class Util
    {
    /* ------------------------------------------------------------------ */
    /* Default prefix string of the namespace.                             */
    /* ------------------------------------------------------------------ */
    private string prefixText = "dnmsamp";

    /* ------------------------------------------------------------------ */
    /* Constructor that takes the replacement prefix string to use.       */
    /* ------------------------------------------------------------------ */
    public Util(String text)
    {
      prefixText = text;
    }

    /* ------------------------------------------------------------------ */
```

```
    /* Display an arbitrary string to the console.                       */
    /* ----------------------------------------------------------------- */
    public void Print(String text)
    {
      System.Console.WriteLine("{0} {1}\n", prefixText, text);
    }



 /* ----------------------------------------------------------------- */
    /* Display the content of the message passed to the console.         */
    /* ----------------------------------------------------------------- */
    public void PrintMessage(MQMessage message)
    {
      if (message.Format.CompareTo(MQC.MQFMT_STRING) == 0)
      {
        try
        {
          string messageText = message.ReadString(message.MessageLength);

          Print(messageText);
        }

        catch(Exception ex)
        {
          Print(ex.ToString());
        }
      }
      else
      {
        Print("UNRECOGNISED FORMAT");
      }
    }

    /* ----------------------------------------------------------------- */
    /* Convert the byte array into a hex string.                         */
    /* ----------------------------------------------------------------- */
    static public string ToHexString(byte[] byteArray)
    {
      string hex = "0123456789ABCDEF";

      string retString = "";

      for(int i = 0; i < byteArray.Length; i++)
      {
        int h = (byteArray[i] & 0xF0)>>4;
        int l = (byteArray[i] & 0x0F);

        retString += hex.Substring(h,1) + hex.Substring(l,1);
      }

      return retString;
    }
  }
}
```

Kompilace programů produktu IBM MQ .NET

Specické příkazy pro kompilaci aplikací .NET zapsaných v různých jazycích.

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Chcete-li sestavit aplikaci C# pomocí produktu IBM MQ classes for .NET, použijte následující příkaz:

```
csc /t:exe /r:System.dll /r:amqmdnet.dll /lib: MQ_INSTALLATION_PATH\bin /out:MyProg.exe
MyProg.cs
```

Chcete-li sestavit aplikaci ve Visual Basicu pomocí produktu IBM MQ classes for .NET, použijte následující příkaz:

```
vbc /r:System.dll /r: MQ_INSTALLATION_PATH\bin\amqmdnet.dll /out:MyProg.exe MyProg.vb
```

Chcete-li sestavit spravovanou aplikaci C++ pomocí produktu IBM MQ classes for .NET, použijte následující příkaz:

```
cl /clr MQ_INSTALLATION_PATH\bin Myprog.cpp
```

Pro ostatní jazyky si prohlédněte dokumentaci dodanou prodejcem jazyka.

Použití samostatného klienta IBM MQ .NET

Klient produktu IBM MQ .NET nabízí od produktu IBM MQ 8.0.0 Fix Pack 2možnost zabalit a implementovat sestavu IBM MQ .NET , aniž by bylo nutné použít úplnou instalaci klienta produktu IBM MQ na produkčních systémech pro spouštění vašich aplikací.

Informace o této úloze

V produktu IBM MQ 8.0.0 Fix Pack 2můžete sestavovat aplikace produktu IBM MQ .NET na počítači, kde je instalován celý klient IBM MQ , a později balík IBM MQ .NET , tj. produkt `amqmdnet.dll`, spolu s aplikací a implementovat jej do produkčních systémů.

Aplikace, které sestavujete a implementujete, mohou být tradičními aplikacemi Windows .NET , službami nebo Microsoft Azure Web/Worker.

V takových implementacích klient produktu IBM MQ .NET podporuje pouze spravovaný režim konektivity ke správci front. Vazby serveru a nespravovaná konektivita režimu klienta nejsou k dispozici, protože tyto dva režimy vyžadují úplnou instalaci klienta IBM MQ . Jakýkoli pokus o použití těchto dvou režimů vede k výjimce aplikace.

Procedura

Odkazování na sestavení klienta IBM MQ .NET v aplikacích

- Ve své aplikaci odkazujte na sestavu `amqmdnet.dll` stejným způsobem jako v předchozích verzích.

  Nastavte vlastnost **`CopyLocal`** sestavy `amqmdnet` na hodnotu `True` , chcete-li zajistit, aby se sestavení `amqmdnet` zkopírovalo do adresáře `bin` aplikace. Nastavení této vlastnosti také pomáhá nástroji pro tvorbu balíků aplikací balit požadované binární soubory pro implementaci do produkčních systémů a také prostředí cloudu Microsoft Azure PaaS .

Přidání podpory globálních transakcí

- Ujistěte se, že aplikace implementuje aplikaci monitoru WMQDotnetXAMonitor na počítači spolu s aplikací samotnou.

  Pokud aplikace používá funkci globální transakce spravované produktem IBM MQ .NET , musí implementovat také WMQDotnetXAMonitor na počítači spolu s aplikací samotnou. Tento obslužný program je potřebný pro zotavení všech sporných transakcí.

Spuštění a zastavení trasování

- Chcete-li spustit a zastavit trasování, použijte konfigurační soubor aplikace a konfigurační soubor trasování specifického pro IBM MQ .

  Poznámka: Následující kroky pro generování trasování se vztahují na opětovně distribuovatelný spravovaný klient produktu .NET a na samostatného klienta .NET .

  Musíte použít konfigurační soubor aplikace a specifický konfigurační soubor trasování produktu IBM MQ , protože, protože neexistuje úplná instalace klienta IBM MQ , nejsou standardní nástroje používané pro spuštění a zastavení trasování, **`strmqtrc`** a **`endmqtrc`**, dostupné.

  Konfigurační soubor aplikace (`app.config` nebo `web.config`)
  Aplikace musí definovat vlastnost **`MQTRACECONFIGFILEPATH`** pod sekcí `<appSettings>` konfiguračního souboru aplikace, to znamená soubor `app.config` nebo `web.config` . (Skutečný název konfiguračního souboru aplikace závisí na názvu vaší aplikace.) Hodnota vlastnosti

MQTRACECONFIGFILEPATH určuje cestu pro umístění specifického konfiguračního souboru trasování IBM MQ , `mqtrace.config`, jak ukazuje následující příklad:

```
<appSettings>
<add key="MQTRACECONFIGFILEPATH" value="C:\MQTRACECONFIG"/>
</appSettings>
```

Trasování je vypnuto, pokud soubor `mqtrace.config` nebyl nalezen v cestě, která je uvedena v konfiguračním souboru aplikace. Nicméně, First Failure Support Technology (FFST) a protokoly chyb jsou vytvořeny v adresáři aplikace, pokud má aplikace oprávnění k zápisu do aktuálního adresáře.

Specifický konfigurační soubor trasování produktu IBM MQ (`mqtrace.config`)

Soubor `mqtrace.config` je soubor XML, který definuje vlastnosti pro spuštění a zastavení trasování, cestu k trasovacím souborům a cestu k protokolům chyb. Tyto vlastnosti jsou popsány v následující tabulce.

Tabulka 78. Vlastnosti definované v souboru `mqtrace.config`

Atribut	Popis
MQTRACELEVEL	0: Zastaví trasování-jedná se o výchozí hodnotu. 1: Spustí trasování s menšími podrobnostmi. 2: Spustí trasování s úplnými podrobnostmi-doporučuje se.
MQTRACEPATH	Ukazuje na složku, kde budou vytvářeny trasovací soubory. Je-li cesta prázdná nebo není definován atribut **MQTRACEPATH** , použije se aktuální adresář aplikace.
MQERRORPATH	Ukazuje na složku, kde budou vytvářeny soubory protokolu chyb. Je-li cesta prázdná nebo není definován atribut **MQERRORPATH** , použije se aktuální adresář aplikace.

Následující příklad zobrazuje ukázkový soubor `mqtrace.config` :

```
<?xml version="1.0" encoding="utf-8"?>
<traceSettings>
  <MQTRACELEVEL>2</MQTRACELEVEL>
  <MQTRACEPATH>C:\MQTRACEPATH</MQTRACEPATH>
  <MQERRORPATH>C:\MQERRORLOGPATH</MQERRORPATH>
</traceSettings>
```

Trasování lze spustit a zastavit dynamicky, je-li aplikace spuštěna, a to změnou hodnoty atributu **MQTRACELEVEL** v souboru `mqtrace.config` .

Spuštěné aplikace musí mít oprávnění k vytvoření a zápisu pro složku určenou atributem **MQTRACELEVEL** pro generování trasovacích souborů. Aplikace, které jsou spuštěny v prostředí produktu Microsoft Azure PaaS , musí také zajistit podobná přístupová oprávnění, protože webové aplikace používající sestavení IBM MQ .NET spuštěné v produktu Microsoft Azure PaaS , nemusejí mít oprávnění k vytváření a zápisu. Generování trasování, zachycení dat prvního selhání (FDC) a protokoly chyb se nezdaří, pokud aplikace nemá požadovaná oprávnění k vytvoření a zápisu pro uvedenou složku.

Povolení přesměrování vazby

- Chcete-li povolit odkaz na vazbu doby kompilace sestavení IBM MQ .NET na novější verzi sestavení, přidejte do konfiguračního souboru aplikace vlastnost <dependentAssembly>.

Následující ukázkový úsek kódu v souboru `app.config` přesměruje aplikaci, která byla kompilována pomocí verze IBM MQ 8.0.0 Fix Pack 2 (`8.0.0.2`) sestavy IBM MQ .NET , ale později byla použita opravná sada IBM MQ 8.0.0 Fix Pack 3, která byla poté použita k aktualizaci IBM MQ.NET sestavení na `8.0.0.3`.

```
<runtime>
  <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
    <!-- amqmdnet related binding redirect -->
    <dependentAssembly>
      <assemblyIdentity name="amqmdnet"
                        publicKeyToken="dd3cb1c9aae9ec97"
                        culture="neutral"/>
      <codeBase version="8.0.0.2"
            href="file:///amqmdnet.dll"/>
      <bindingRedirect oldVersion="1.0.0.3-8.0.0.2"
                       newVersion="8.0.0.3"/>
      <publisherPolicy apply="no"/>
    </dependentAssembly>
  </assemblyBinding>
</runtime>
```

Související pojmy

“Použití aplikace WMQDotnetXAMonitor” na stránce 528
Aplikace WMQDotnetXAMonitor musí být spuštěna ručně. Může být spuštěn kdykoliv. Můžete jej spustit, když se zobrazí zprávy v systému SYSTEM.DOTNET.XARECOVERY.QUEUE nebo ji můžete nechat běžet na pozadí před tím, než provedete libovolnou transakční práci s aplikacemi, které jsou zapsány pomocí tříd produktu IBM MQ .NET .

Související informace

Komponenty a funkce produktu IBM MQ

Redistribuovatelné klienty

Běhové prostředí aplikace .NET - Windows pouze

Použití rozhraní Model objektu Component Model (IBM MQ Automation Classes for ActiveX)

Produkt IBM MQ Automation Classes for ActiveX (MQAX) jsou komponenty ActiveX , které poskytují třídy, které můžete použít ve své aplikaci pro přístup k produktu IBM MQ.

V 9.0.0 V produktu IBM MQ 9.0je podpora produktu MicrosoftActive X zamítnuta. Třídy IBM MQ pro .NET jsou doporučenými náhradními technologiemi. Další informace viz Vývoj aplikací .NET.

Produkt MQAX vyžaduje prostředí produktu IBM MQ a odpovídající aplikaci produktu IBM MQ , se kterou má komunikovat.

Poskytuje vám aplikaci ActiveX schopnost spouštět transakce a přistupovat k datům na libovolném z vašich podnikových systémů, ke kterým můžete přistupovat prostřednictvím produktu IBM MQ.

IBM MQ Automation Classes for ActiveX:

- Poskytnutí přístupu k funkcím a funkcím rozhraní API produktu IBM MQ umožňuje úplnou vzájemnou konektivitu s jinými platformami IBM MQ .
- Přepočítá se s normálními konvencemi, které se očekávají od komponenty ActiveX .
- Conform to the IBM MQ object model, also available for .NET, C + +, Java, and LotusScript.

Jsou poskytnuty ukázky spouštěče MQAX. Tyto ukázky můžete použít nejprve ke kontrole, zda je instalace produktu MQAX úspěšná a zda máte základní prostředí produktu IBM MQ na místě. Ukázky také ukazují, jak lze použít MQAX.

Skriptování COM a ActiveX

The Component Object Model (COM) is an object-based programming model defined by Microsoft. Uvádí, jak mohou být softwarové komponenty poskytovány způsobem, který jim umožňuje lokalizovat a komunikovat mezi sebou bez ohledu na jazyk počítače, ve kterém jsou napsány, nebo na jejich umístění.

ActiveX je sada technologií založených na COM, které integruje vývoj aplikací, znovupoužitelné komponenty a internetové technologie na platformách Microsoft Windows . Komponenty ActiveX poskytují rozhraní, ke kterým lze dynamicky přistupovat pomocí aplikací. Skriptovací klient ActiveX je aplikace, například kompilátor, který může sestavit nebo spustit program nebo skript, který používá rozhraní poskytovaná komponentami ActiveX (nebo COM).

Podpora prostředí produktu IBM MQ

IBM MQ Automation Classes for ActiveX lze použít pouze s **32bitovými** skriptovacími klienty ActiveX .

Komponentu COM lze použít pouze pro aplikace **32-bit** . Chcete-li zapsat 64bitovou aplikaci COM, můžete použít rozhraní produktu .NET .

Chcete-li spustit volání MQAX v prostředí serveru IBM MQ , musíte mít v systému nainstalován produkt Windows 2000 nebo novější.

To run the MQAX in an IBM MQ MQI client environment you need IBM MQ MQI client on Windows 2000 or later installed on your system:

Produkt IBM MQ MQI client vyžaduje přístup k alespoň jednomu serveru IBM MQ . Pokud jsou ve vašem systému nainstalovány oba servery IBM MQ MQI client i server IBM MQ , budou aplikace MQAX vždy spuštěny vůči serveru. Rozhraní ActiveX k rozhraní MQAI je k dispozici pouze v prostředí serveru IBM MQ .

Návrh a programování pomocí IBM MQ Automation Classes for ActiveX

Návrh aplikací MQAX, které přistupují k jiným aplikacím nežActiveX

Třídy automatizace produktu IBM MQ poskytují přístup k funkcím rozhraní API produktu IBM MQ . Proto můžete využívat výhody ze všech výhod, které produkt IBM MQ může přinést do vaší aplikace Windows .

Celkový návrh vaší aplikace je stejný jako u všech aplikací produktu IBM MQ , takže zvažte všechny aspekty návrhu popsané v sekci “Aspekty návrhu pro aplikace produktu IBM MQ” na stránce 43 .

Chcete-li používat třídy automatizace IBM MQ , programujete programy produktu Windows ve vaší aplikaci pomocí jazyka, který podporuje vytváření a používání objektů COM. Příklad: Visual Basic, Javaa další skriptovací klienti ActiveX . Třídy lze poté snadno integrovat do vaší aplikace, protože objekty produktu IBM MQ , které potřebujete, mohou být kódovány pomocí nativní syntaxe jazyka implementace.

Použití Automatizačních tříd produktu IBM MQ pro ActiveX

Při návrhu aplikace ActiveX , která používá IBM MQ Automation Classes for ActiveX, je nejdůležitější položkou informací zpráva, která je odeslána nebo přijata ze vzdáleného systému IBM MQ . Proto musíte znát formát položek, které jsou vloženy do zprávy. Pro skript MQAX k práci musí znát strukturu zprávy i aplikace produktu IBM MQ , která tuto zprávu vyzvedne nebo odešle.

Pokud odesíláte zprávu s aplikací MQAX a chcete provést převod dat na konci struktury MQAX, musíte také vědět:

- Kódová stránka použitá vzdáleným systémem
- Kódování použité vzdáleným systémem

Chcete-li uchovat svůj kód přenosný, je dobrým zvykem nastavit kódovou stránku a kódování, i když jsou v současné době stejné jak v odesílajícím, tak v přijímajícím systému.

Při zvažování, jak strukturovat implementaci systému, který navrhujete, pamatujte na to, že se skripty MQAX spouštějí na stejném počítači jako ten, na kterém je nainstalován buď správce front produktu IBM MQ , nebo klient IBM MQ .

Rady a tipy pro programování

Následující rady a tipy nejsou ve významném pořadí. Jsou to témata, která vám mohou ušetřit čas, pokud jsou důležitá pro práci, kterou děláte.

Vlastnosti deskriptoru zpráv

Pokud manipulujete s vlastnostmi deskriptoru zpráv v programu, může být lepší použít hexadecimální ekvivalenty polí.

Informace v této části odkazují na následující vlastnosti:

- AccountingToken
- CorrelationId
- GroupId
- MessageId

Pokud je aplikace produktu IBM MQ původcem zprávy a produkt IBM MQ tyto vlastnosti vygeneruje, je lepší použít vlastnosti AccountingTokenHex, CorrelationIdHex, GroupIdHex a MessageIdHex, pokud se chcete podívat na jejich hodnoty nebo s nimi pracovat jakýmkoli způsobem, včetně jejich předání ve zprávě do produktu IBM MQ. Důvodem pro to je, že generované hodnoty IBM MQ jsou řetězce bajtů, které mají libovolnou hodnotu od 0 do 255 včetně, nejsou to řetězce tisknutelných znaků.

Kde je váš skript MQAX původcem zprávy, můžete použít buď vlastnosti AccountingToken, CorrelationId, GroupIda MessageId nebo jejich hexadecimální ekvivalenty.

IBM MQ konstanty

IBM MQ konstanty jsou k dispozici jako členové výčtu IBM MQ v knihovně MQAX200.

řetězcové konstanty řetězce IBM MQ

Řetězcové konstanty IBM MQ nejsou k dispozici, když používáte IBM MQ Automation Classes for ActiveX. Musíte použít explicitní znakový řetězec pro ty, které jsou uvedeny v následujícím seznamu a všechny ostatní, které byste mohli potřebovat. Tyto příkazy musí být doplněny na osm znaků pomocí mezer:

Tabulka 79. Řetězcové konstanty IBM MQ a jejich odpovídající znakové řetězce.

Řetězcová konstanta	**Odpovídající znakový řetězec**
MQFMT_NONE	" "
MQFMT_ADMIN	"MQADMIN"
MQFMT_CHANNEL_COMPLETED	"MQCHCOM"
MQFMT_CICS	"MQCICS"
MQFMT_COMMAND_1	"MQCMD1 "
MQFMT_COMMAND_2	"MQCMD2 "
HLAVIČKA MQFMT_DEAD_LETTER_HEADER	"MQDEAD"
ZÁHLAVÍ MQFMT_DICT_HEADER	"MQHDIST"
UDÁLOST MQFMT_EVENT	"MQEVENT"
MQFMT_IMS	"MQIMS"
MQFMT_IMS_VAR_STRING	"MQIMSVS"
ROZŠÍŘENÍ MQFMT_MD_EXTENSION	"MQHMDE"
MQFMT_PCF	"MQPCF"

Tabulka 79. Řetězcové konstanty IBM MQ a jejich odpovídající znakové řetězce. (pokračování)	
Řetězcová konstanta	**Odpovídající znakový řetězec**
MQFMT_REF_MSG_HEADER	"MQHREF"
ZÁHLAVÍ MQFMT_RF_HEADER	"MQHRF"
ŘETĚZEC MQFMT_STRING	"MQSTR"
SPOUŠTĚČ MQFMT_TRIGGER	"MQTRIG"
MQFMT_WORK_INFO_HEADER	"MQHWIH"
ZÁHLAVÍ MQFMT_XMIT_Q_HEADER	"MQXMIT"

Konstanty řetězce null

Konstanty IBM MQ použité pro inicializaci čtyř vlastností MQMessage, MQMI_NONE (24 NULL znaků), MQCI_NONE (24 NULL znaků), MQGI_NONE (24 NULL znaků) a MQACT_NONE (32 NULL znaků), nejsou podporovány třídami automatizace IBM MQ pro ActiveX. Nastavení na prázdné řetězce má stejný efekt.

Chcete-li například nastavit různá ID zprávy MQMessage na tyto hodnoty: *mymessage*. **MessageId** = "" *mymessage*. **CorrelationId** = "" *mymessage*. **AccountingToken** = ""

Příjem zprávy z produktu IBM MQ

Existuje několik způsobů přijetí zprávy z produktu IBM MQ:

- Systém výzev pomocí funkce GET následovaný funkcí TIMER s použitím funkce Vizuální Basic TIMER.
- Zadání volby GET s volbou Wait; zadáte dobu čekání nastavením vlastnosti WaitInterval . Zvažte tuto možnost, i když jste nastavili systém, aby se spouštěl v prostředí s více vlákny, software, který běží v daném okamžiku, může běžet pouze s jedním vláknem. To zabrání tomu, aby se systém zamyká neomezeně.

  Ostatní vlákna nejsou ovlivněna. Pokud však ostatní podprocesy vyžadují přístup k produktu IBM MQ, vyžadují druhé připojení k produktu IBM MQ pomocí dalších objektů správce front MQAX a objektů front.

Zadání příkazu GET s volbou Wait a nastavením parametru WaitInterval na MQWI_UNLIMITED způsobí, že se systém zamkne až do dokončení volání GET, pokud je proces jediným vláknem.

Použití konverze dat

Dvě formy převodu dat jsou podporovány IBM MQ Automation Classes for ActiveX -číselné kódování a konverze znakové sady.

Numerické kódování

Nastavíte-li vlastnost MQMessage Encoding, následující metody se převedou mezi různými systémy kódování čísel:

- Metoda ReadDecimal2
- Metoda ReadDecimal4
- Metoda ReadDouble
- Metoda ReadDouble4
- Metoda ReadFloat
- Metoda ReadInt2
- Metoda ReadInt4
- Metoda ReadLong
- Metoda ReadShort

- Metoda ReadUInt2
- Metoda WriteDecimal2
- Metoda WriteDecimal4
- Metoda WriteDouble
- Metoda WriteDouble4
- Metoda WriteFloat
- Metoda WriteInt2
- Metoda WriteInt4
- Metoda WriteLong
- Metoda WriteShort
- Metoda WriteUInt2

Vlastnost Encoding může být nastavena a interpretována pomocí dodaných konstant IBM MQ . Obrázek 58 na stránce 565 zobrazuje příklad těchto:

```
/* Encodings for Binary Integers */
MQENC_INTEGER_UNDEFINED
MQENC_INTEGER_NORMAL
MQENC_INTEGER_REVERSED

/* Encodings for Decimals */
MQENC_DECIMAL_UNDEFINED
MQENC_DECIMAL_NORMAL
MQENC_DECIMAL_REVERSED

/* Encodings for Floating-Point Numbers */
MQENC_FLOAT_UNDEFINED
MQENC_FLOAT_IEEE_NORMAL
MQENC_FLOAT_IEEE_REVERSED
MQENC_FLOAT_S390
```

Obrázek 58. Dodávané IBM MQ konstanty pro kódování

Chcete-li například odeslat celé číslo ze systému Intel do operačního systému System/390 v kódování System/390 , postupujte takto:

```
Dim msg As New MQMessage  'Define an IBM MQ message for our use..
Print msg. Encoding   'Currently 546 (or X'222')
                                          'Set the encoding property
                                           to 785 (or X'311')
msg. Encoding = MQENC_INTEGER_NORMAL OR MQENC_DECIMAL_NORMAL
                          OR MQENC_FLOAT_S390
Print msg. Encoding   'Print it to see the change
Dim local_num As long 'Define a long integer
local_num = 1234                          'Set it
msg. WriteLong (local_num)       'Write the number into the message
```

Převod znakové sady

Konverze znakové sady je nezbytná, když posíláte zprávu z jednoho systému do jiného systému, kde jsou kódové stránky odlišné. Převod kódové stránky se používá:

- Metoda ReadString
- Metoda ReadNullTerminatedString
- Metoda WriteString
- Metoda WriteNullTerminatedString
- Vlastnost MessageData

Vlastnost MQMessage CharacterSet musí být nastavena na podporovanou hodnotu znakové sady (CCSID).

Produkt IBM MQ Automation Classes for ActiveX používá převodní tabulky k provedení konverze znakové sady.

Chcete-li například převést řetězce automaticky na kódovou stránku 437:

```
Dim msg As New MQMessage                'Define an IBM MQ message
msg.CharacterSet = 437                 'Set code page required
msg.WriteString "A character string"'Put character string in message
```

Metoda WriteString přijímá řetězcová data (`A character string` v příkladu) jako řetězec Unicode. Pak převede tato data z Unicode do kódové stránky 437 pomocí převodní tabulky 34B001B5.TBL.

Znaky v řetězci Unicode, které nejsou podporovány kódovou stránkou 437, jsou na kódové stránce 437 poskytnuty standardnímu substitučním znaku.

Podobně, když použijete metodu ReadString , příchozí zpráva má znakovou sadu vytvořenou pomocí hodnoty MQMD (IBM MQ Message Descriptor) a před předáním zpět do skriptovacího jazyka převod z této kódové stránky do Unicode.

Využití vláken

IBM MQ Automation Classes for ActiveX implementují model s podporou podprocesů, ve kterém mohou být objekty používány mezi podprocesy.

Přestože produkt MQAX povoluje použití objektů MQQueue a MQQueueManager , produkt IBM MQ v současné době nepovoluje sdílení manipulátorů mezi různými podprocesy.

Pokusy o použití těchto objektů na jiném podprocesu způsobí chybu a produkt IBM MQ vrátí návratový kód MQRC_HCONN_ERROR.

Poznámka: Pro každý proces existuje pouze jeden objekt MQSession. Použití relace MQSession CompletionCode a ReasonCode se nedoporučuje ve vícevláknových prostředích. Hodnoty chyb MQSession mohou být přepsány druhým vláknem mezi chybou, která byla vyvolána a kontrolována na prvním vláknu. Podprocesy jsou serializovány po dobu trvání každého volání metody nebo přístupu k vlastnosti. Takže zadáním příkazu Get s volbou Wait způsobí, že další podprocesy, které přistupují k objektům MQAX, budou pozastaveny, dokud nebude operace dokončena.

Ošetření chyb

Tyto informace popisují vlastnosti objektu MQAX, jak ošetřování chyb funguje, pravidla popisující způsob zpracování výjimek a získání vlastnosti.

Každý objekt MQAX obsahuje vlastnosti obsahující informace o chybě a metodu resetování nebo vymazání těchto vlastností. Vlastnosti jsou:

- CompletionCode
- ReasonCode
- ReasonName

Metoda je:

- Kódy ClearError

Jak funguje ošetřování chyb

Váš skript nebo aplikace MQAX vyvolá metodu objektu MQAX nebo přístupy nebo aktualizace vlastnosti objektu MQAX:

1. Aktualizují se kódy ReasonCode a CompletionCode v příslušném objektu.
2. Aktualizují se také ReasonCode a CompletionCode v objektu MQSession se stejnými informacemi.

   Poznámka: Omezení týkající se použití chybových kódů MQSession v aplikacích s podporou podprocesů naleznete v příručce “Využití vláken” na stránce 566 .

Je-li kód CompletionCode větší nebo roven vlastnosti ExceptionThreshold objektu MQSession, funkce MQAX vyvolá výjimku (číslo 32000). Použijte jej v rámci skriptu pomocí příkazu On-Chyba (nebo ekvivalentního příkazu), který má být zpracován.

3. Pomocí funkce Error můžete načíst přidružený chybový řetězec, který má tvar:

```
MQAX: CompletionCode=xxx, ReasonCode=xxx, ReasonName=xxx
```

Další informace o tom, jak používat chybové příkazy, naleznete v dokumentaci k skriptovacímu jazyku ActiveX .

Použití položky CompletionCode a ReasonCode v objektu MQSession je vhodné pro jednoduché obslužné rutiny chyb.

Vlastnost ReasonName vrací symbolický název IBM MQ pro aktuální hodnotu prvku ReasonCode.

Vyvolání výjimek

Následující pravidla popisují, jak se zachází s výjimkami:

- Kdykoli vlastnost nebo metoda nastaví kód dokončení na hodnotu větší než nebo rovnou prahové hodnotě výjimky (obvykle je nastavena na 2), dojde k výjimce.
- Všechny volání metod a sady vlastností nastavují kód dokončení.

Získání vlastnosti

Jedná se o speciální případ, protože CompletionCode a ReasonCode nejsou vždy aktualizovány:

- Je-li vlastnost úspěšná, objekt a objekt MQSession ReasonCode a CompletionCode zůstávají nezměněny.
- Pokud vlastnost get selže s hodnotou CompletionCode varování, zůstane ReasonCode a CompletionCode nezměněno.
- Pokud vlastnost get selže s chybou CompletionCode , aktualizují se hodnoty ReasonCode a CompletionCode tak, aby odrážely skutečné hodnoty, a zpracování chyb bude pokračovat podle popisu.

Třída MQSession má metodu *ReasonCodeName* , která může být použita k nahrazení kódu příčiny IBM MQ symbolickým názvem. To je obzvláště užitečné při vývoji programů, v nichž může dojít k neočekávaným chybám. Název však není ideální pro prezentaci uživatelům.

Každá třída má také vlastnost *ReasonName*, která vrací symbolický název aktuálního kódu příčiny pro danou třídu.

IBM MQ Odkaz na třídy automatizace pro ActiveX

Tento oddíl popisuje třídy produktu IBM MQ Automation Classes for ActiveX (MQAX), vyvinuté pro ActiveX. Třídy umožňují psát aplikace ActiveX , které mohou přistupovat k dalším aplikacím spuštěným v jiných prostředích nežActiveX , pomocí produktu IBM MQ.

Rozhraní IBM MQ Automation Classes for ActiveX

Produkt IBM MQ Automation Classes for ActiveX poskytuje předdefinované číselné konstanty ActiveX (například MQMT_REQUEST), které jsou potřebné pro použití tříd.

Automatizační třídy ActiveX se skládají z následujících položek:

- “Třída MQSession” na stránce 569
- “Třída MQQueueManager” na stránce 572
- “Třída MQQueue” na stránce 583
- “Třída MQMessage” na stránce 598

- “Třída voleb MQPutMessage” na stránce 621
- “Třída voleb MQGetMessage” na stránce 623
- “Třída MQDistributionList” na stránce 625
- “Třída položek MQDistributionList” na stránce 630

Kromě toho produkt IBM MQ Automation Classes for ActiveX poskytuje předdefinované číselné konstanty ActiveX (jako např. MQMT_REQUEST), které jsou potřebné pro použití tříd. Ty jsou poskytovány ve výčtu MQ v knihovně MQAX200. Konstanty jsou podmnožinou proměnných definovaných v souborech záhlaví IBM MQ C (cmqc * .h) s některými dalšími třídami automatizace IBM MQ pro kódy příčiny ActiveX .

O třídách automatizace IBM MQ pro třídy ActiveX

Přečtěte si tyto informace spolu s referenčními tématy v části Vývoj odkazů na aplikace.

See Features that can be used only with the primary installation on Windows for important information.

Třída MQSession poskytuje kořenový objekt, který obsahuje stav poslední akce provedené na kterémkoli z objektů MQAX. Další informace viz “Ošetření chyb” na stránce 566.

Třídy MQQueueManager a MQQueue poskytují přístup k základním objektům produktu IBM MQ . Metody nebo přístupy vlastností pro tyto třídy obecně vedou k volání v rámci IBM MQ MQI.

Třídy voleb MQMessage, MQPutMessagea MQGetMessagezapouzdřují datové struktury MQMD, MQPMO a MQGMO a používají se jako pomůcce při odesílání zpráv do front a načítání zpráv z nich.

Třída MQDistributionList zapouzdřuje kolekci front-local, remote nebo alias pro výstup. Třída položek MQDistributionListzapouzdřuje struktury MQOR, MQRR a MQPMR a přidruží je k seznamu distribučních distribučních položek.

Předání parametru

Parametry při vyvolání metody jsou předávány hodnotou, kromě případů, kdy tento parametr je objekt, a v tom případě je předáván odkaz.

Poskytnutý seznam definic tříd uvádí datový typ pro každý parametr nebo vlastnost. Pro mnoho klientů ActiveX , jako je Visual Basic, pokud použitá proměnná není požadovaného typu, je hodnota automaticky převedena na nebo z požadovaného typu-poskytující takový převod je možný. Tato pravidla se řídí standardními pravidly klienta; MQAX takový převod neposkytuje.

Mnohé z metod používají řetězcové parametry pevné délky nebo vracejí znakový řetězec pevné délky. Převodní pravidla jsou následující:

- Pokud uživatel zadá řetězec s pevnou délkou o chybné délce, jako vstupní parametr nebo jako návratovou hodnotu, hodnota je oseknuta nebo doplněna koncovými mezerami, jak je požadováno.
- Pokud uživatel zadá řetězec s proměnnou délkou o nesprávné délce jako vstupní parametr, je hodnota oříznuta nebo doplněna koncovými mezerami.
- Pokud uživatel zadá řetězec proměnné délky chybné délky jako návratovou hodnotu, řetězec se upraví na požadovanou délku (protože vrací hodnotu, zničí předchozí hodnotu v řetězci i tak).
- Řetězce poskytnuté jako vstupní parametry mohou obsahovat vložené hodnoty Null.

Tyto třídy lze nalézt v knihovně MQAX200 .

Metody přístupu k objektu

Tyto metody se nevztahují přímo k žádnému volání funkce IBM MQ . Každá z těchto metod vytvoří objekt, ve kterém jsou umístěny referenční informace, následovány připojením k objektu IBM MQ nebo jeho otevřením:

Je-li vytvořeno připojení ke správci front, uchovává atribut 'manipulátor připojení' generovaný produktem IBM MQ.

Když je fronta otevřena, uchovává atribut 'object handle' generovaný produktem IBM MQ.

Tyto atributy produktu IBM MQ nejsou přímo k dispozici pro program MQAX.

Chyby

Syntaktické chyby při předávání parametru mohou být zjištěny v době kompilace a běhové prostředí klientem ActiveX . Chyby mohou být zachyceny pomocí Chyba při chybě ve Visual Basic.

Všechny třídy IBM MQ ActiveX obsahují dvě speciální vlastnosti jen pro čtení- ReasonCode a CompletionCode. Tyto vlastnosti lze číst kdykoli.

Pokus o přístup k jakékoli jiné vlastnosti, nebo k vydání jakéhokoli volání metody může generovat chybu z IBM MQ.

Je-li vlastnost nebo vyvolání metody úspěšné, je hodnota ReasonCode vlastnícího objektu nastavena na hodnotu MQRC_NONE a položka CompletionCode je nastavena na hodnotu MQCC_OK.

Pokud není přístup k vlastnosti nebo vyvolání metody úspěšný, jsou v těchto polích nastaveny kódy příčiny a dokončení.

Třída MQSession

Toto je kořenová třída produktu IBM MQ Automation Classes for ActiveX.

Pro klientský proces ActiveX vždy existuje pouze jeden objekt MQSession. Při pokusu o vytvoření druhého objektu se vytvoří druhý odkaz na původní objekt.

VYTVOŘENÍ

Volba **Nový** vytvoří nový objekt MQSession.

Syntaxe

Dim *mqsess* **Jako nový MQSession Nastavit** *mqsess* = **Nová relace MQSession**

Vlastnosti

- “Vlastnost CompletionCode” na stránce 569.
- “Vlastnost ExceptionThreshold” na stránce 570.
- “Vlastnost ReasonCode” na stránce 570.
- “Vlastnost ReasonName” na stránce 571.

Metoda

- “Metoda AccessGetMessageOptions” na stránce 571.
- “Metoda AccessMessage” na stránce 571.
- “Metoda AccessPutMessageOptions” na stránce 571.
- “Metoda AccessQueueManager” na stránce 571.
- “Metoda ClearErrorCodes” na stránce 572.
- “Metoda názvu ReasonCode” na stránce 572.

Vlastnost CompletionCode

Pouze pro čtení. Vrací kód dokončení IBM MQ nastavený nejnovější metodou nebo sadou vlastností, která byla vydána na libovolném objektu IBM MQ .

Při úspěšném vyvolání metody nebo sady vlastností pro kterýkoli objekt MQAX je obnovení metody MQCC_OK resetováno.

Obslužná rutina události chyby může zkontrolovat tuto vlastnost a diagnostikovat chybu, aniž by bylo nutné vědět, který objekt byl zahrnut.

Použití CompletionCode a ReasonCode v objektu MQSession je velmi výhodné pro jednoduché obslužné rutiny chyb.

Poznámka: Omezení týkající se použití chybových kódů MQSession v aplikacích s podporou podprocesů naleznete v příručce “Využití vláken” na stránce 566 .

Definováno v:
Třída MQSession

Datový typ:
Dlouhý

Hodnoty:

- MQCC_OK
- VAROVÁNÍ MQCC_WARNING
- SELHÁNÍ MQCC_FAILED

Syntaxe:
Chcete-li získat: *completioncode & = MQSession*.CompletionCode

Vlastnost ExceptionThreshold

Čtení a zápis. Definuje úroveň chyby IBM MQ , pro kterou operace MQAX vygeneruje výjimku. Výchozí hodnota je MQCC_FAILED. Hodnota větší než MQCC_FAILED účinně zabraňuje zpracování výjimek, takže programátor může provést kontroly na CompletionCode a ReasonCode.

Definováno v: Třída MQSession

Datový typ: Long

Hodnoty:

- Jakýkoli, ale zvažte MQCC_WARNING, MQCC_FAILED nebo vyšší.

Syntaxe:

Chcete-li získat následující informace: *ExceptionThreshold& = MQSession*. **ExceptionThreshold**

Nastavení: *MQSession.* **ExceptionThreshold** = *ExceptionThreshold$*

Vlastnost ReasonCode

Pouze pro čtení. Vrací kód příčiny nastavený nejnovější metodou nebo sadou vlastností, která byla vydána pro každý objekt IBM MQ .

Obslužná rutina události chyby může zkontrolovat tuto vlastnost a diagnostikovat chybu, aniž by bylo nutné vědět, který objekt byl zahrnut.

Použití CompletionCode a ReasonCode v objektu MQSession je velmi výhodné pro jednoduché obslužné rutiny chyb.

Poznámka: Omezení týkající se použití chybových kódů MQSession v aplikacích s podporou podprocesů naleznete v příručce “Využití vláken” na stránce 566 .

Definováno v: Třída MQSession

Datový typ: Long

Hodnoty:

- Viz Příčina (MQLONG) a další hodnoty MQAX uvedené v části “Kód příčiny IBM MQ Automation Classes for ActiveX .” na stránce 637.

Syntaxe: Chcete-li získat: *reasoncode & = MQSession*. **ReasonCode**

Vlastnost ReasonName

Pouze pro čtení. Vrátí symbolický název posledního kódu příčiny. Příklad: "MQRC_QMGR_NOT_AVAILABLE".

Poznámka: Omezení týkající se použití chybových kódů MQSession v aplikacích s podporou podprocesů naleznete v příručce “Využití vláken” na stránce 566 .

Definováno v: Třída MQSession

Datový typ: Řetězec

Hodnoty:

- Viz termín completion API a kódy příčin.

Syntaxe: Chcete-li získat: *reasonname $= MQSession* **.ReasonName**

Metoda AccessGetMessageOptions

Vytvoří nový objekt voleb MQGetMessage.

Definováno v:
Třída MQSession

Syntaxe:
gmo = MQSession **.AccessGetMessageOptions()**

Metoda AccessMessage

Vytvoří nový objekt MQMessage.

Definováno v:
Třída MQSession

Syntaxe:
msg = MQSession **.AccessMessage()**

Metoda AccessPutMessageOptions

Vytvoří nový objekt voleb MQPutMessage.

Definováno v:
Třída MQSession

Syntaxe:
pmo = MQSession **.AccessPutMessageOptions()**

Metoda AccessQueueManager

Vytvoří nový objekt MQQueueManager a připojí jej ke skutečnému správci front prostřednictvím serveru IBM MQ MQI client nebo IBM MQ . Kromě provádění připojení tato metoda také provádí otevření pro objekt správce front.

Když jsou ve vašem systému nainstalovány oba servery IBM MQ MQI client a IBM MQ , aplikace MQAX se standardně spouštějí na serveru. Chcete-li spustit produkt MQAX pro klienta, musí být v proměnné prostředí `GMQ_MQ_LIB` určena knihovna vazeb klienta, například nastavte `GMQ_MQ_LIB=mqic.dll`.

V případě instalace pouze klienta není nutné nastavit proměnnou prostředí `GMQ_MQ_LIB` . Není-li tato proměnná nastavena, produkt IBM MQ se pokusí o načtení souboru amqzst.dll. Pokud tato knihovna DLL není přítomna (jak je tomu v případě instalace klienta), IBM MQ se pokusí zavést soubor mqic.dll.

Je-li funkce úspěšná, nastaví hodnotu ConnectionStatus produktu MQQueueManagerna hodnotu TRUE.

Správce front může být připojen k nejvýše jednomu objektu MQQueueManager na instanci ActiveX .

Dojde-li k selhání připojení ke správci front, dojde k vyvolání chybové události a k nastavení objektu MQSession ReasonCode a CompletionCode .

Definováno v: Třída MQSession

Syntaxe: *set qm = MQSession* **.AccessQueueManager** (*Name$*)

Parametr: *Název$* Řetězec. Název správce front, ke kterému má být připojen.

Metoda ClearErrorCodes

Resetuje parametr CompletionCode na MQCC_OK a ReasonCode na MQRC_NONE.

Definováno v: Třída MQSession

Syntaxe:

```
Call MQSession.ClearErrorCodes()
```

Metoda názvu ReasonCode

Vrátí název kódu příčiny s danou číselnou hodnotou. Je užitečné poskytnout jasnější indikaci chybových stavů uživatelům. Název je stále poněkud zašifrováno (například ReasonCodeName (2059) je **MQRC_Q_MGR_NOT_AVAILABLE**), takže případné chyby by měly být zachyceny a nahrazeny popisným textem odpovídajícím aplikaci.

Definováno v: Třída MQSession

Syntaxe: *errname $= MQSession* **.ReasonCodeName** (*reasonCode&*)

Parametr: *reasoncode &* Long. Kód příčiny, pro který je vyžadován symbolický název.

Třída MQQueueManager

Tato třída představuje připojení ke správci front. Správce front může být spuštěn lokálně (server IBM MQ) nebo vzdáleně s přístupem poskytovaného klientem IBM MQ . Aplikace musí vytvořit objekt této třídy a připojit jej ke správci front. Je-li objekt z této třídy zničen, je automaticky odpojen od správce front.

Obsažení

Objekty třídy MQQueue jsou přidruženy k této třídě.

Nový vytvoří nový objekt MQQueueManager a nastaví všechny vlastnosti na počáteční hodnoty. Alternativně použijte metodu AccessQueueManager třídy MQSession.

VYTVOŘENÍ

Nový vytvoří objekt **nový** MQQueueManager a nastaví všechny vlastnosti na počáteční hodnoty. Alternativně použijte metodu AccessQueueManager třídy MQSession.

Syntaxe

Dim *mgr* **As New MQQueueManager set** *mgr* = **New MQQueueManager**

Vlastnosti

- “Vlastnost ID AlternateUser” na stránce 574.
- “Vlastnost AuthorityEvent” na stránce 574.
- “Vlastnost BeginOptions” na stránce 574.
- “Vlastnost definice ChannelAutoDefinition” na stránce 575.
- “Vlastnost ChannelAutoDefinitionEvent” na stránce 575.
- “Vlastnost ChannelAutoDefinitionExit” na stránce 575.
- “Vlastnost CharacterSet” na stránce 575.

- “Vlastnost CloseOptions” na stránce 575.
- “Vlastnost CommandInputvlastnostQueueName” na stránce 576.
- “Vlastnost CommandLevel” na stránce 576.
- “Vlastnost CompletionCode” na stránce 576.
- “Vlastnost ConnectionHandle” na stránce 576.
- “Vlastnost ConnectionStatus” na stránce 576.
- “Vlastnost ConnectOptions” na stránce 577.
- “Vlastnost DeadLetterQueueName” na stránce 577.
- “Vlastnost DefaultTransmissionQueueName” na stránce 577.
- “Vlastnost popisu” na stránce 577.
- “Vlastnost DistributionLists” na stránce 577.
- “Vlastnost InhibitEvent” na stránce 578.
- “Vlastnost IsConnected” na stránce 578.
- “Vlastnost IsOpen” na stránce 578.
- “Vlastnost LocalEvent” na stránce 578.
- “Vlastnost MaximumHandles” na stránce 578.
- “Vlastnost MaximumMessageLength” na stránce 579.
- “Vlastnost MaximumPriority” na stránce 579.
- “Vlastnost zpráv MaximumUncommitted” na stránce 579.
- “Vlastnost name” na stránce 579.
- “Vlastnost ObjectHandle” na stránce 579.
- “Vlastnost PerformanceEvent” na stránce 579.
- “Vlastnost platformy” na stránce 580.
- “Vlastnost ReasonCode” na stránce 580.
- “Vlastnost ReasonName” na stránce 580.
- “Vlastnost RemoteEvent” na stránce 580.
- “Vlastnost události StartStop” na stránce 581.
- “Vlastnost Dostupnost SyncPoint” na stránce 581.
- “Vlastnost TriggerInterval” na stránce 581.

Metody

- “Metoda AccessQueue” na stránce 581.
- “Metoda seznamu AddDistribution” na stránce 582.
- “Metoda vrácení” na stránce 582.
- “Metoda Begin” na stránce 582.
- “Metoda ClearErrorCodes” na stránce 582.
- “Metoda Commit” na stránce 583.
- “Metoda Connect” na stránce 583.
- “Metoda Disconnect” na stránce 583.

Přístup k majetku

K následujícím vlastnostem lze přistupovat kdykoli.

- “Vlastnost ID AlternateUser” na stránce 574.

- “Vlastnost CompletionCode” na stránce 576.
- “Vlastnost ConnectionStatus” na stránce 576.
- “Vlastnost ReasonCode” na stránce 580.

Ke zbývajícím vlastnostem lze přistupovat pouze v případě, že je objekt připojen ke správci front, a ID uživatele je autorizováno pro zjišťování proti tomuto správci front. Je-li nastaveno alternativní ID uživatele a aktuální ID uživatele je oprávněno používat, alternativní ID uživatele se kontroluje místo toho autorizace pro dotaz.

Pokud tyto podmínky neplatí, produkt IBM MQ Automation Classes for ActiveX se pokusí připojit ke správci front a otevřít jej pro zjištění automaticky. Není-li tento výsledek úspěšný, volání nastaví CompletionCode funkce MQCC_FAILED a jeden z následujících kódů příčiny ReasonCodes:

- PORCC_CONNECTION_CONNECTION_LO
- AUTORIZOVANÝ MQRC_NOT_AUTHORIZED
- CHYBA MQRC_Q_MGR_NAME_ERROR
- MQRC_Q_MGR_NOT_AVAILABLE

Vlastnost ID AlternateUser

Čtení a zápis. Alternativní ID uživatele, které má být použito pro ověření přístupu k atributům správce front.

Tato vlastnost nesmí být nastavena, je-li hodnota IsConnected nastavena na hodnotu TRUE.

Tuto vlastnost nelze nastavit, když je objekt otevřený.

Třída **Defined in:** MQQueueManager

Data Type: Řetězec o délce 12 znaků

Syntax: Chcete-li získat: *altuser $= MQQueueManager* **.AlternateUserId** K nastavení: *MQQueueManager* **.AlternateUserId** = *altuser $*

Vlastnost AuthorityEvent

Pouze pro čtení. Atribut AuthorityEvent rozhraní MQI.

Definováno v:
Třída MQQueueManager

Datový typ:
Dlouhý

Hodnoty:

- MQEV_DISABLED
- POVOLENÝ MQEVR_

Syntaxe: Chcete-li získat: *authevent = MQQueueManager* **.AuthorityEvent**

Vlastnost BeginOptions

Čtení a zápis. Jedná se o volby, které se použijí na metodu Begin. Původně MQBO_NONE.

Definováno v:
Třída MQQueueManager

Datový typ:
Dlouhý

Hodnoty:

- MQBO_NONE

Syntaxe: Chcete-li získat: *beginoptions & =MQQueueManager*. **BeginOptions**

Pro nastavení: *MQQueueManager* **.BeginOptions** = *beginoptions &*

Vlastnost definice ChannelAutoDefinition

Pouze pro čtení. Tento ovládací prvek určuje, zda je povolena automatická definice kanálu.

Definováno v:
Třída MQQueueManager

Datový typ:
Dlouhý

Hodnoty:

- MQCHAD_DISABLED
- MQCHAD_ENABLED

Syntaxe: Chcete-li získat: *channelautodef & = MQQueueManager*. **ChannelAuto**

Vlastnost ChannelAutoDefinitionEvent

Pouze pro čtení. Tento ovládací prvek určuje, zda jsou generovány události s automatickou definicí kanálu.

Definováno v:
Třída MQQueueManager

Datový typ:
Dlouhý

Hodnoty:

- MQEV_DISABLED
- POVOLENÝ MQEVR_

Syntaxe: Chcete-li získat: *channelautodefevent & =MQQueueManager*. **ChannelAutoDefinitionEvent**

Vlastnost ChannelAutoDefinitionExit

Pouze pro čtení. Název uživatelské procedury použité pro automatickou definici kanálu.

Definováno v:
Třída MQQueueManager

Datový typ:
Řetězec

Syntaxe: Chcete-li získat následující informace: *channelautodefexit$= MQQueueManager*.
ChannelAutoDefinitionExit

Vlastnost CharacterSet

Pouze pro čtení. Atribut CodedCharSetId rozhraní MQI.

Definováno v: třídě MQQueueManager

Datový typ: Long

Syntaxe: Chcete-li získat: *characterset & = MQQueueManager* **.CharacterSet**

Vlastnost CloseOptions

Čtení a zápis. Volby používané k řízení toho, co se stane, když je správce front uzavřen. Počáteční hodnota je MQCO_NONE.

Definováno v:
Třída MQQueueManager

Datový typ:
Dlouhý

Hodnoty:

- MQCO_NONE

Syntaxe: Chcete-li získat následující informace: *closeopt & = MQQueueManager* **.CloseOptions**

Pro nastavení: *MQQueueManager* **.CloseOptions** = *closeopt &*

Vlastnost CommandInputvlastnostQueueName

Pouze pro čtení. Atribut QName CommandInputMQI.

Definováno v: třídě MQQueueManager

Typ dat: Řetězec 48 znaků

Syntaxe: Chcete-li získat: *commandinputqname $= MQQueueManager* **.CommandInputQueueName**

Vlastnost CommandLevel

Pouze pro čtení. Vrací verzi a úroveň implementace správce front produktu IBM MQ (atribut MQI CommandLevel)

Definováno v: třídě MQQueueManager

Datový typ: Long

Syntaxe: Chcete-li získat: *level & = MQQueueManager* **.CommandLevel**

Vlastnost CompletionCode

Pouze pro čtení. Vrací kód dokončení nastavený podle poslední metody nebo přístupu k vlastnosti, který byl vydán proti objektu.

Definováno v: třídě MQQueueManager

Datový typ: Long

Hodnoty:

- MQCC_OK
- VAROVÁNÍ MQCC_WARNING
- SELHÁNÍ MQCC_FAILED

Syntaxe: Chcete-li získat: *completioncode & = MQQueueManager* **.CompletionCode**

Vlastnost ConnectionHandle

Pouze pro čtení. Popisovač připojení pro objekt správce front produktu IBM MQ .

Definováno v:
Třída MQQueueManager

Datový typ:
Dlouhý

Syntaxe: Chcete-li získat následující informace: *hconn & = MQQueueManager.* **ConnectionHandle**

Vlastnost ConnectionStatus

Pouze pro čtení. Indikuje, zda je objekt připojen ke svému správci front, nebo ne.

Definováno v: třídě MQQueueManager

Typ dat: Boolean

Hodnoty:

- TRUE (-1)
- FALSE (0)

Syntaxe: Chcete-li získat: *status = MQQueueManager* **.ConnectionStatus**

Vlastnost ConnectOptions

Čtení a zápis. Tyto volby se týkají metody Connect. Zpočátku MQCNO_NONE.

Definováno v:
Třída MQQueueManager

Datový typ:
Dlouhý

Hodnoty:

- VAZBA MQCNO_STANDARD_BINDING
- VAZBA MQCNO_FASTPATH_BINDING
- MQCNO_NONE

Syntaxe: Chcete-li získat: *connecttoptions & =MQQueueManager*. **ConnectOptions**

Pro nastavení: *MQQueueManager* **.ConnectOptions** = *connecttoptions &*

Vlastnost DeadLetterQueueName

Pouze pro čtení. Atribut QName DeadLetterrozhraní MQI.

Definováno v: třídě MQQueueManager

Typ dat: Řetězec 48 znaků

Syntaxe: pro získání: *dlqname $= MQQueueManager* **.DeadLetterQueueName**

Vlastnost DefaultTransmissionQueueName

Pouze pro čtení. Atribut QName DefXmitMQI.

Definováno v: třídě MQQueueManager

Typ dat: Řetězec 48 znaků

Syntaxe: Chcete-li získat: *defxmitqname $= MQQueueManager* **.DefaultTransmissionQueueName**

Vlastnost popisu

Pouze pro čtení. Atribut QMgrDesc rozhraní MQI.

Definováno v: třídě MQQueueManager

Typ dat: Řetězec o délce 64 znaků

Syntaxe: Chcete-li získat: *description $= MQQueueManager* **.Popis**

Vlastnost DistributionLists

Pouze pro čtení. Jedná se o schopnost správce front podporovat distribuční seznamy.

Definováno v:
Třída MQQueueManager

Datový typ:
Logická hodnota

Hodnoty:

- TRUE (-1)
- FALSE (0)

Syntaxe: Chcete-li získat: *distributionlists= MQQueueManager*. **DistributionLists**

Vlastnost InhibitEvent

Pouze pro čtení. Atribut MQI InhibitEvent .

Definováno v: třídě MQQueueManager

Datový typ: Long

Hodnoty:

- MQEV_DISABLED
- POVOLENÝ MQEVR_

Syntaxe: Chcete-li získat: *inhibevent & = MQQueueManager* **.InhibitEvent**

Vlastnost IsConnected

Hodnota, která označuje, zda je správce front momentálně připojen.

Pouze pro čtení.

Definováno v: třídě MQQueueManager

Typ dat: Boolean

Hodnoty:

- TRUE (-1)
- FALSE (0)

Syntaxe: Chcete-li získat: *isconnected = MQQueueManager* **.IsConnected**

Vlastnost IsOpen

Hodnota, která označuje, zda je správce front aktuálně otevřen pro zjišťování.

Pouze pro čtení.

Definováno v:
Třída MQQueueManager

Datový typ:
Logická hodnota

Hodnoty:

- TRUE (-1)
- FALSE (0)

Syntaxe: Chcete-li získat: *IsOpen = MQQueueManager*. **IsOpen**

Vlastnost LocalEvent

Pouze pro čtení. Atribut LocalEvent rozhraní MQI.

Definováno v: třídě MQQueueManager

Datový typ: Long

Hodnoty:

- MQEV_DISABLED
- POVOLENÝ MQEVR_

Syntaxe: Chcete-li získat: *localevent & = MQQueueManager* **.LocalEvent**

Vlastnost MaximumHandles

Pouze pro čtení. Atribut MaxHandles MQI.

Definováno v: třídě MQQueueManager

Datový typ: Long

Syntaxe: Chcete-li získat: *maxhandles & = MQQueueManager* **.MaximumHandles**

Vlastnost MaximumMessageLength

Pouze pro čtení. Atribut MaxMsgQueue Manager MQI.

Definováno v: třídě MQQueueManager

Datový typ: Long

Syntaxe: Chcete-li získat následující informace: *maxmessagelength & = MQQueueManager* **.MaximumMessageLength**

Vlastnost MaximumPriority

Pouze pro čtení. Atribut MaxPriority rozhraní MQI.

Definováno v: třídě MQQueueManager

Datový typ: Long

Syntaxe: Chcete-li získat: *maxpriority & = MQQueueManager* **.MaximumPriority**

Vlastnost zpráv MaximumUncommitted

Pouze pro čtení. Atribut MaxUncommittedzpráv rozhraní MQI.

Definováno v: třídě MQQueueManager

Datový typ: Long

Syntaxe: Chcete-li získat následující informace: *maxuncommitted & = MQQueueManager* **.MaximumUncommittedMessages**

Vlastnost name

Čtení a zápis. Atribut QMgrName rozhraní MQI (MQI). Tuto vlastnost nelze zapsat, jakmile je připojen objekt MQQueueManager .

Definováno v: třídě MQQueueManager

Typ dat: Řetězec 48 znaků

Syntaxe: Chcete-li získat: *name $= MQQueueManager* **.name**

Chcete-li nastavit: *MQQueueManager* **.name** = *název $*

Poznámka: Visual Basic rezervuje vlastnost "Name" pro použití ve vizuálním rozhraní. Proto při použití ve Visual Basic use lower-case, tj. "name".

Vlastnost ObjectHandle

Pouze pro čtení. Popisovač objektu pro objekt správce front produktu IBM MQ .

Definováno v:
Třída MQQueueManager

Datový typ
Dlouhý

Syntaxe: Chcete-li získat: *hobj & = MQQueueManager.* **ObjectHandle**

Vlastnost PerformanceEvent

Pouze pro čtení. Atribut PerformanceEvent rozhraní MQI.

Definováno v: třídě MQQueueManager

Datový typ: Long

Hodnoty:

- MQEV_DISABLED
- POVOLENÝ MQEVR_

Syntaxe: Chcete-li získat: *perfevent & = MQQueueManager*.PerformanceEvent

Vlastnost platformy

Pouze pro čtení. Atribut platformy MQI.

Definováno v: třídě MQQueueManager

Datový typ: Long

Hodnoty:

- POČ MQPL_WINDOWS_NT
- OKNA MQPL_WINDOWS

Syntaxe: Chcete-li získat: *platform & = MQQueueManager* **.Platforma**

Vlastnost ReasonCode

Pouze pro čtení. Vrátí kód příčiny nastavený posledním voláním metody nebo vlastností, které byly vydány pro daný objekt.

Definováno v: třídě MQQueueManager

Datový typ: Long

Hodnoty:

- Viz termín completion API a kódy příčin.

Syntaxe: Chcete-li získat následující informace: *reasoncode & = MQQueueManager* **.ReasonCode**

Vlastnost ReasonName

Pouze pro čtení. Vrátí symbolický název posledního kódu příčiny. Příklad: "MQRC_QMGR_NOT_AVAILABLE".

Definováno v: třídě MQQueueManager

Datový typ: Řetězec

Hodnoty:

- Viz termín completion API a kódy příčin.

Syntaxe: Chcete-li získat následující informace: *reasonname $= MQQueueManager* **.ReasonName**

Vlastnost RemoteEvent

Pouze pro čtení. Atribut RemoteEvent rozhraní MQI.

Definováno v: třídě MQQueueManager

Datový typ: Long

Hodnoty:

- MQEV_DISABLED
- POVOLENÝ MQEVR_

Syntaxe: Chcete-li získat: *remoteevent & = MQQueueManager* **.RemoteEvent**

Vlastnost události StartStop

Pouze pro čtení. Atribut události rozhraní MQI StartStop.

Definováno v: třídě MQQueueManager

Datový typ: Long

Hodnoty:

- MQEV_DISABLED
- POVOLENÝ MQEVR_

Syntaxe: Chcete-li získat: *strstpevent & = MQQueueManager* **.StartStopEvent**

Vlastnost Dostupnost SyncPoint

Pouze pro čtení. Atribut SyncPoint rozhraní MQI.

Definováno v: třídě MQQueueManager

Datový typ: Long

Hodnoty:

- MQSP_AVAILABLE
- MQSP_NOT_AVAILABLE

Syntaxe: Chcete-li získat: *syncpointailability & = MQQueueManager* **.SyncPointAvailability**

Vlastnost TriggerInterval

Pouze pro čtení. Atribut TriggerInterval MQI.

Definováno v: třídě MQQueueManager

Datový typ: Long

Syntaxe: Chcete-li získat: *trigint & = MQQueueManager* **.TriggerInterval**

Metoda AccessQueue

Vytvoří objekt MQQueue a přidruží jej k tomuto objektu MQQueueManager nastavením vlastnosti odkazu na připojení fronty. Nastavuje vlastnosti Název, OpenOptions, DynamicQueuea AlternateUserID objektu MQQueue na zadané hodnoty a pokusí se ji otevřít.

Je-li otevření neúspěšné, volání selže. U objektu je vyvolána chybová událost. Jsou nastaveny volby ReasonCode a CompletionCodea MQSession ReasonCode a CompletionCode objektu.

Parametry DynamicQueue, QueueManagera AlternateUserID jsou volitelné a výchozí hodnoty "".

Pokud mají být načteny vlastnosti fronty, měly by být kromě jiných voleb určeny také parametry OpenOption MQO_INQUIRE.

Nenastavujte název objektu QueueManagernebo jej nastavte na hodnotu "", je-li fronta, která má být otevřena, lokální. Jinak jej nastavte na název vzdáleného správce front, který je vlastníkem fronty, a dojde k pokusu o otevření lokální definice vzdálené fronty. Další informace o rozlišování názvů vzdálených front a aliasing správce front naleznete v tématu Co jsou aliasy?.

Je-li vlastnost Název nastavena na název modelové fronty, zadejte název dynamické fronty, která má být vytvořena, v parametru DynamicQueueName$. Je-li hodnota zadaná v parametru DynamicQueueName$ nastavena na hodnotu "", hodnota nastavená na objekt fronty a použitá v otevřeném volání je "AMQ. *". Další informace o pojmenovávání dynamických front naleznete v příručce “Vytváření dynamických front” na stránce 721 .

Definice

Definováno v: Třída MQQueueManager .

Syntaxe

Syntaxe: set queue = MQQueueManager. **AccessQueue** (Name$, OpenOptions&, QueueManagerName$, DynamicQueueName$, AlternateUserId$)

Parametry

Název$ Řetězec. Název fronty produktu IBM MQ .

OpenOptions: Long. Volby, které mají být použity při otevření fronty. Viz OpenOptions (MQLONG).

Řetězec *QueueManagerName$* . Název správce front, který vlastní frontu, jež má být otevřena. Hodnota "" znamená, že správce front je lokální.

Řetězec *DynamicQueueName$* . Název přiřazený k dynamické frontě v době, kdy je fronta otevřena, když parametr Name$ uvádí modelovou frontu.

AlternateUserId$ String. Alternativní ID uživatele použité k ověření přístupu při otevření fronty.

Metoda seznamu AddDistribution

Vytvoří nový objekt MQDistributionList a nastaví svůj odkaz na připojení na vlastníka správce front.

Definováno v:
Třída MQQueueManager

Syntaxe: *set distributtionlist* = **MQQueueManager.** *SeznamAddDistribution*

Metoda vrácení

Zálohuje všechny nepotvrzené zprávy typu put a gets, které se vyskytly jako část transakce od posledního synchronizačního bodu.

Definováno v: třídě MQQueueManager

Syntaxe:

```
Call MQQueueManager.Backout()
```

Metoda Begin

Začne jednotku práce, která je koordinována správcem front. Volby zahájení ovlivňují chování této metody.

Definováno v:
Třída MQQueueManager

Syntaxe:

```
Call MQQueueManager.Begin()
```

Metoda ClearErrorCodes

Resetuje parametr CompletionCode na MQCC_OK a ReasonCode na MQRC_NONE jak pro třídu MQQueueManager , tak pro třídu MQSession.

Definováno v:
Třída MQQueueManager

Syntaxe:
Volání *MQQueueManager* **.ClearErrorCodes ()**

Metoda Commit

Potvrzuje všechny zprávy put a gets, které se vyskytly jako součást pracovní jednotky od posledního synchronizačního bodu.

Definováno v: třídě MQQueueManager

Syntaxe:

```
Call MQQueueManager.Commit()
```

Metoda Connect

Připojí objekt MQQueueManager ke skutečnému správci front prostřednictvím produktu IBM MQ MQI client nebo serveru. Stejně jako při vytváření připojení tato metoda také otevře objekt správce front, aby mohl být dotazován.

Nastaví IsConnected na TRUE.

Pro připojení ke správci front je povoleno maximálně jeden objekt MQQueueManager pro instanci ActiveX .

Definováno v: třídě MQQueueManager

Syntaxe:

```
Call MQQueueManager.Connect()
```

Metoda Disconnect

Odpojí objekt MQQueueManager od správce front.

Nastaví hodnotu IsConnected na hodnotu FALSE.

Všechny objekty fronty přidružené k objektu MQQueueManager jsou nepoužitelné a nelze je znovu otevřít.

Veškeré nepotvrzené změny (operace vložení a získávání zpráv) jsou potvrzeny.

Definováno v: třídě MQQueueManager

Syntaxe:

```
Call MQQueueManager.Disconnect()
```

Třída MQQueue

Tato třída představuje přístup k frontě produktu IBM MQ . Toto připojení je poskytováno přidruženým objektem MQQueueManager . Je-li objekt z této třídy zničen, automaticky se zavře.

Obsažení

Třída MQQueue je obsažena ve třídě MQQueueManager .

VYTVOŘENÍ

Produkt New vytvoří nový objekt MQQueue a nastaví všechny vlastnosti na počáteční hodnoty. Případně můžete použít metodu AccessQueue třídy MQQueueManager .

Syntaxe

```
Dim que As New MQQueue Set que = New MQQueue
```

Vlastnosti

- “Vlastnost ReasonName” na stránce 593.
- “Vlastnost RemoteQueueManagerName” na stránce 594.
- “Vlastnost názvu RemoteQueueName” na stránce 594.
- “Vlastnost ResolvedQueueManagerName” na stránce 594.
- “Vlastnost názvu ResolvedQueue” na stránce 594.
- “Vlastnost RetentionInterval” na stránce 594.
- “Vlastnost scope” na stránce 594.
- “Vlastnost ServiceInterval” na stránce 595.
- “Vlastnost události ServiceInterval” na stránce 595.
- “Vlastnost Shareability” na stránce 595.
- “Vlastnost názvu TransmissionQueue” na stránce 595.
- “Vlastnost TriggerControl” na stránce 595.
- “Vlastnost TriggerData” na stránce 595.
- “Vlastnost TriggerDepth” na stránce 596.
- “Vlastnost priority TriggerMessage” na stránce 596.
- “Vlastnost TriggerType” na stránce 596.
- “vlastnost použití” na stránce 596.

Metody

- “Metoda ClearErrorCodes” na stránce 596
- “Metoda Close” na stránce 597
- “metoda GET” na stránce 597
- “Otevřít metodu” na stránce 598
- “metoda PUT” na stránce 598

Přístup k majetku

Není-li objekt fronty připojen ke správci front, můžete si přečíst následující vlastnosti:

- “Vlastnost CompletionCode” na stránce 587
- “Vlastnost OpenStatus” na stránce 592
- “Vlastnost ReasonCode” na stránce 593

a můžete číst a zapisovat do:

- “Vlastnost ID AlternateUser” na stránce 586
- “Vlastnost CloseOptions” na stránce 587
- “Vlastnost ConnectionReference” na stránce 587
- “Vlastnost name” na stránce 591
- “Vlastnost OpenOptions” na stránce 592

Je-li objekt fronty připojen ke správci front, můžete si přečíst všechny vlastnosti.

Vlastnosti atributu fronty

Vlastnosti, které nejsou uvedeny v předchozí sekci, jsou všechny atributy základní fronty IBM MQ . K nim lze přistupovat pouze tehdy, je-li objekt připojen ke správci front a jméno uživatele uživatele je autorizováno pro zjišťování nebo sadu proti této frontě. Je-li nastaveno alternativní ID uživatele a k jeho použití je autorizované aktuální ID uživatele, zkontroluje se místo toho alternativní ID uživatele.

Vlastnost musí být vhodnou vlastností pro daný typ QueueType. Další informace naleznete v tématu Atributy pro fronty .

Pokud tyto podmínky neplatí, přístup k vlastnosti nastaví parametr CompletionCode MQCC_FAILED a jeden z následujících ReasonCodes:

- PORCC_CONNECTION_CONNECTION_LO
- AUTORIZOVANÝ MQRC_NOT_AUTHORIZED
- CHYBA MQRC_Q_MGR_NAME_ERROR
- PŘIPOJENÍ MQRC_Q_MGR_NOT_CONNECTED
- MQRC_SELECTOR_NOT_FOR_TYPE (CompletionCode je MQCC_WARNING)

Otevření fronty

Jediným způsobem, jak vytvořit objekt MQQueue, je použití metody AccessQueue MQQueueManager nebo pomocí volby Nový. Otevřený objekt MQQueue zůstane otevřený (OpenStatus= TRUE), dokud nebude odstraněn nebo odstraněn, nebo dokud nebude objekt správce front odstraněn nebo pokud se ztratí připojení ke správci front. Hodnota vlastnosti MQQueue CloseOptions řídí chování operace zavření, ke které dojde, když je objekt MQQueue odstraněn.

Metoda MQQueueManager AccessQueue otevírá frontu pomocí parametru OpenOptions . Metoda MQQueue.Open otevře frontu s použitím vlastnosti OpenOptions . Produkt IBM MQ ověřuje platnost OpenOptions proti autorizaci uživatele jako součásti procesu otevřené fronty.

Vlastnost ID AlternateUser

Čtení a zápis. Alternativní ID uživatele použité pro ověření přístupu k frontě při jeho otevření.

Tuto vlastnost nelze nastavit při otevření objektu (to znamená, že pokud je položka IsOpen TRUE).

Definováno v: Třída MQQueue

Datový typ: Řetězec o délce 12 znaků

Syntaxe: Chcete-li získat: *altuser $= MQQueue* **.AlternateUserId**

Nastavení: *MQQueue.* **AlternateUserId** = *altuser $*

Vlastnost názvu BackoutRequeue

Pouze pro čtení. Atribut BackOutRequeueQName MQI.

Definováno v: Třída MQQueue

Typ dat: Řetězec 48 znaků

Syntaxe: Chcete-li získat následující informace: *backoutrequeuename $= MQQueue* **.BackoutRequeueName**

Vlastnost BackoutThreshold

Pouze pro čtení. Atribut BackoutThreshold rozhraní MQI.

Definováno v: Třída MQQueue

Datový typ: Long

Hodnoty:

- Viz BackoutThreshold (MQLONG).

Syntaxe: Chcete-li získat: *backoutthreshold & = MQQueue.* **BackoutThreshold**

Vlastnost názvu BaseQueue

Pouze pro čtení. Název fronty, na kterou je alias vyřešen.

Platné pouze pro alias fronty.

Definováno v: Třída MQQueue

Typ dat: Řetězec 48 znaků

Syntaxe: Chcete-li získat: *baseqname $= MQQueue* **.BaseQueueName**

Vlastnost CloseOptions

Čtení a zápis. Volby používané k řízení toho, co se stane, když je fronta zavřena.

Definováno v: Třída MQQueue

Datový typ: Long

Hodnoty:

- MQCO_NONE
- MQCO_DELETE
- MQCO_DELETE_PURGE

MQCO_DELETE a MQCO_DELETE_PURGE jsou platné pouze pro dynamické fronty.

Syntaxe: Chcete-li získat: *closeopt & = MQQueue* **.CloseOptions**

Chcete-li nastavit: *MQQueue* **.CloseOptions** = *closeopt &*

Vlastnost CompletionCode

Pouze pro čtení. Vrací kód dokončení nastavený podle poslední metody nebo přístupu k vlastnosti, který byl vydán proti objektu.

Definováno v: Třída MQQueue

Datový typ: Long

Hodnoty:

- MQCC_OK
- VAROVÁNÍ MQCC_WARNING
- SELHÁNÍ MQCC_FAILED

Syntaxe: Chcete-li získat: *completioncode & = MQQueue* **.CompletionCode**

Vlastnost ConnectionReference

Čtení a zápis. Definuje objekt správce front, do kterého náleží objekt fronty. Odkaz na připojení nelze zapsat, když je fronta otevřená.

Definováno v: Třída MQQueue

Datový typ: MQQueueManager

Hodnoty:

- Odkaz na aktivní objekt správce front produktu IBM MQ

Syntaxe: Chcete-li nastavit: *set MQQueue* **.ConnectionReference** = *ConnectionReference*

Chcete-li získat následující informace: *set ConnectionReference = MQQueue* **.ConnectionReference**

Vlastnost Time CreationDate

Pouze pro čtení. Datum a čas vytvoření této fronty.

Definováno v: Třída MQQueue

Typ dat: Varianta typu 7 (datum/čas) nebo EMPTY

Syntaxe: Chcete-li získat: *datetime = MQQueue* **.CreationDateTime**

Vlastnost CurrentDepth

Pouze pro čtení. Počet zpráv, které se právě nacházejí ve frontě.

Definováno v: Třída MQQueue

Datový typ: Long

Syntaxe: Chcete-li získat: *currentdepth & = MQQueue* **.CurrentDepth**

Vlastnost DefaultInputOpenOption

Pouze pro čtení. Řídí způsob, jakým je fronta otevřena, pokud OpenOptions uvádí MQOO_INPUT_AS_Q_DEF.

Definováno v: Třída MQQueue

Datový typ: Long

Hodnoty:

- MQO_INPUT_EXCLUSIVE
- MQO_INPUT_SHARED

Syntaxe: Chcete-li získat: *defaultinop & = MQQueue* **.DefaultInputOpenOption**

Vlastnost DefaultPersistence

Pouze pro čtení. Výchozí trvání pro zprávy ve frontě.

Definováno v: Třída MQQueue

Datový typ: Long

Syntaxe: Chcete-li získat: *defpersistence & = MQQueue* **.DefaultPersistence**

Vlastnost DefaultPriority

Pouze pro čtení. Výchozí priorita pro zprávy ve frontě.

Definováno v: Třída MQQueue

Datový typ: Long

Syntaxe: Chcete-li získat: *defPriority & = MQQueue* **.DefaultPriority**

Vlastnost DefinitionType

Pouze pro čtení. Typ definice fronty.

Definováno v: Třída MQQueue

Datový typ: Long

Hodnoty:

- MQQDT_PREDEFINED
- MQQDT_PERMANENT_DYNAMIC
- MQQDT_DOČASNÝ_DYNAMICKÝ

Syntaxe: Chcete-li získat: *deftype & = MQQueue* **.DefinitionType**

DepthHigh-vlastnost události

Pouze pro čtení. Atribut události rozhraní MQI QDepthHigh.

Definováno v: Třída MQQueue

Datový typ: Long

Hodnoty:

- MQEV_DISABLED
- POVOLENÝ MQEVR_

Syntaxe: Chcete-li získat: *depthhighevent & = MQQueue.* **DepthHighUdálost**

DepthHighOmezit vlastnost

Pouze pro čtení. Atribut Omezení QDepthHighMQI.

Definováno v: Třída MQQueue

Datový typ: Long

Syntaxe: Chcete-li získat: *depthhighlimit & = MQQueue.* **DepthHighLimit**

DepthLow-vlastnost události

Pouze pro čtení. Atribut události rozhraní MQI QDepthLow.

Definováno v: Třída MQQueue

Datový typ: Long

Hodnoty:

- MQEV_DISABLED
- POVOLENÝ MQEVR_

Syntaxe: Chcete-li získat: *depthlowevent & = MQQueue.* **DepthLowUdálost**

DepthLow-vlastnosti omezení

Pouze pro čtení. Atribut Omezení QDepthLowMQI.

Definováno v: Třída MQQueue

Datový typ: Long

Syntaxe: Chcete-li získat: *depthlowlimit & = MQQueue.* **DepthLow-limit**

DepthMaximumVlastnost události

Pouze pro čtení. Atribut události rozhraní MQI QDepthMax.

Definováno v: Třída MQQueue

Datový typ: Long

Hodnoty:

- MQEV_DISABLED
- POVOLENÝ MQEVR_

Syntaxe: Chcete-li získat: *depthmaximumevent & = MQQueue.* **DepthMaximumUdálost**

Vlastnost popisu

Pouze pro čtení. Popis fronty.

Definováno v: Třída MQQueue

Typ dat: Řetězec o délce 64 znaků

Syntaxe: Chcete-li získat: *description $= MQQueue* **.Popis**

Vlastnost názvu DynamicQueue

Čtení a zápis, pouze pro čtení, je-li fronta otevřená.

Tento parametr řídí název dynamické fronty použité při otevření modelové fronty. Může být nastaven jako zástupný znak uživatelem jako sada vlastností (pouze v případě, že je fronta uzavřena) nebo jako parametr pro MQQueueManager.AccessQueue().

Skutečný název dynamické fronty se nachází dotazem QueueName.

Definováno v: Třída MQQueue

Typ dat: Řetězec 48 znaků

Hodnoty:

- Libovolné platné jméno fronty IBM MQ .

Syntaxe: Chcete-li nastavit: *MQQueue* **.DynamicQueueName** = *dynamickqueuename $*

Chcete-li získat: *dynamicqueuename $ = MQQueue* **.DynamicQueueName**

Vlastnost HardenGetBackout

Pouze pro čtení. Zda se má udržovat přesný zpětný počet.

Definováno v: Třída MQQueue

Datový typ: Long

Hodnoty:

- MACKY_BACKOUT_HARDENED
- MQQA_BACKOUT_NOT HARDENED

Syntaxe: Chcete-li získat: *hardengetback & = MQQueue* **.HardenGetBackout**

Vlastnost InhibitGet

Čtení a zápis. Atribut InhibitGet rozhraní MQI.

Definováno v: Třída MQQueue

Datový typ: Long

Hodnoty:

- MQQA_GET_INHIBED
- MQQA_GET_ALLOWED

Syntaxe: Chcete-li získat: *getstatus & = MQQueue* **.InhibitGet**

Pro nastavení: *MQQueue* **.InhibitGet** = *getstatus &*

Vlastnost InhibitPut

Čtení a zápis. Atribut MQI InhibitPut .

Definováno v: Třída MQQueue

Datový typ: Long

Hodnoty:

- MQQA_PUT_BLOKOVÁNO
- MQQA_PUT_ALLOWED

Syntaxe: Chcete-li získat: *putstatus & = MQQueue* **.InhibitPut**

Pro nastavení: *MQQueue* **.InhibitPut** = *putstatus &*

Vlastnost názvu InitiationQueue

Pouze pro čtení. Název inicializační fronty.

Definováno v: Třída MQQueue

Typ dat: Řetězec 48 znaků

Syntaxe: Chcete-li získat: *initqname $= MQQueue* **.InitiationQueueName**

Vlastnost IsOpen

Vrátí informaci o tom, zda je fronta otevřená.

Pouze pro čtení.

Definováno v: Třída MQQueue

Typ dat: Boolean

Hodnoty:

- TRUE (-1)
- FALSE (0)

Syntaxe: Chcete-li získat: *open = MQQueue* **.IsOpen**

Vlastnost MaximumDepth

Pouze pro čtení. Maximální hloubka fronty.

Definováno v: Třída MQQueue

Datový typ: Long

Syntaxe: Chcete-li získat: *maxdepth & = MQQueue* **.MaximumDepth**

Vlastnost MaximumMessageLength

Pouze pro čtení. Maximální povolená délka zprávy v bajtech pro tuto frontu.

Definováno v: Třída MQQueue

Datový typ: Long

Syntaxe: Chcete-li získat: *maxmlength & = MQQueue* **.MaximumMessageLength**

Vlastnost posloupnosti MessageDelivery

Pouze pro čtení. Sekvence doručení zpráv.

Definováno v: Třída MQQueue

Datový typ: Long

Hodnoty:

- PRIORITA MQMS_PRIORITY
- MQSD_FIFO

Syntaxe: Chcete-li získat: *messagedeseq & = MQQueue* **.MessageDeliverySequence**

Vlastnost name

Čtení a zápis. Atribut Fronta MQI. Tuto vlastnost nelze zapsat poté, co je otevřena fronta MQQueue.

Definováno v: Třída MQQueue

Typ dat: Řetězec 48 znaků

Syntaxe: Chcete-li získat: *name $= MQQueue* **.name**

Chcete-li nastavit: *MQQueue* **.name** = *název $*

Poznámka: Visual Basic rezervuje vlastnost "Name" pro použití ve vizuálním rozhraní. Proto při použití ve Visual Basic use lower-case, to je "name".

Vlastnost ObjectHandle

Pouze pro čtení. Popisovač objektu pro objekt fronty IBM MQ .

Definováno v: Třída MQQueue

Datový typ: Long

Syntaxe: Chcete-li získat: *hobj & = MQQueue.* **ObjectHandle**

Vlastnost OpenInputCount

Pouze pro čtení. Počet otevření pro vstup.

Definováno v: Třída MQQueue

Datový typ: Long

Syntaxe: Chcete-li získat:

```
openincount& = MQQueue.OpenInputCount
```

Vlastnost OpenOptions

Čtení a zápis. Volby, které mají být použity pro otevření fronty.

Definováno v: Třída MQQueue

Datový typ: Long

Hodnoty:

- Viz OpenOptions (MQLONG).

Syntaxe: Chcete-li získat:

```
openopt& = MQQueue.OpenOptions
```

Nastavení: *MQQueue.* **OpenOptions** = *openopt &*

Vlastnost počtu OpenOutput

Pouze pro čtení. Počet operací otevření pro výstup.

Definováno v: Třída MQQueue

Datový typ: Long

Syntaxe: Chcete-li získat:

```
openoutcount& = MQQueue.OpenOutputCount
```

Vlastnost OpenStatus

Pouze pro čtení. Označuje, zda je fronta otevřena či nikoli. Počáteční hodnota je TRUE po metodě AccessQueue nebo FALSE po Nové.

Definováno v: Třída MQQueue

Typ dat: Boolean

Hodnoty:

- TRUE (-1)

- FALSE (0)

Syntaxe: Chcete-li získat:

```
status& = MQQueue.OpenStatus
```

Vlastnost ProcessName

Pouze pro čtení. Atribut MQI ProcessName .

Definováno v: Třída MQQueue

Typ dat: Řetězec 48 znaků

Syntaxe: Chcete-li získat: *procname $ = MQQueue* **.ProcessName**

Vlastnost názvu QueueManager

Čtení a zápis. Název správce front produktu IBM MQ .

Definováno v: Třída MQQueue

Datový typ: Řetězec

Syntaxe: Chcete-li získat následující informace: *QueueManagerName$ = MQQueue* **.QueueManagerName**

Chcete-li nastavit: *MQQueue* **.QueueManagerName** = *QueueManagerName$*

Vlastnost QueueType

Pouze pro čtení. Atribut QType rozhraní MQI.

Definováno v: Třída MQQueue

Datový typ: Long

Hodnoty:

- ALIAS MQQ_ALIAS
- MQQ_LOCAL
- MQQ_MODEL
- MQQT_REMOTE

Syntaxe: Chcete-li získat: *queuetype & = MQQueue* **.QueueType**

Vlastnost ReasonCode

Pouze pro čtení. Vrátí kód příčiny nastavený posledním voláním metody nebo vlastností, které byly vydány pro daný objekt.

Definováno v: Třída MQQueue

Datový typ: Long

Hodnoty:

- Viz termín completion API a kódy příčin.

Syntaxe: Chcete-li získat: *reasoncode & = MQQueue* **.ReasonCode**

Vlastnost ReasonName

Pouze pro čtení. Vrátí symbolický název posledního kódu příčiny. Příklad: "MQRC_QMGR_NOT_AVAILABLE".

Definováno v: Třída MQQueue

Datový typ: Řetězec

Hodnoty:

- Viz termín completion API a kódy příčin.

Syntaxe: Chcete-li získat: *reasonname $= MQQueue* **.ReasonName**

Vlastnost RemoteQueueManagerName

Pouze pro čtení. Název vzdáleného správce front. Platí pouze pro vzdálené fronty.

Definováno v: Třída MQQueue

Typ dat: Řetězec 48 znaků

Syntaxe: Chcete-li získat: *remqmanname $= MQQueue* **.RemoteQueueManagerName**

Vlastnost názvu RemoteQueueName

Pouze pro čtení. Název fronty, jak je znám ve vzdáleném správci front. Platí pouze pro vzdálené fronty.

Definováno v: Třída MQQueue

Typ dat: Řetězec 48 znaků

Syntaxe: Chcete-li získat: *remqname $= MQQueue* **.RemoteQueueName**

Vlastnost ResolvedQueueManagerName

Pouze pro čtení. Název konečného cílového správce front, jak je znám pro lokálního správce front.

Definováno v: Třída MQQueue

Typ dat: Řetězec 48 znaků

Syntaxe: Chcete-li získat: *resqmanname $= MQQueue* **.ResolvedQueueManagerName**

Vlastnost názvu ResolvedQueue

Pouze pro čtení. Název konečné cílové fronty, jak je znám pro lokálního správce front.

Definováno v: Třída MQQueue

Typ dat: Řetězec 48 znaků

Syntaxe: Chcete-li získat: *reqname $= MQQueue* **.ResolvedQueueNázev**

Vlastnost RetentionInterval

Pouze pro čtení. Doba, po kterou by měla být fronta zadržena.

Definováno v: Třída MQQueue

Datový typ: Long

Syntaxe: Chcete-li získat: *retinterval & = MQQueue* **.RetentionInterval**

Vlastnost scope

Pouze pro čtení. Určuje, zda položka pro tuto frontu také existuje v adresáři buňky.

Definováno v: Třída MQQueue

Datový typ: Long

Hodnoty:

- MQSCOM_Q_MGR
- BUŇKA MQSCO_CELL

Syntaxe: Chcete-li získat: *scope & = MQQueue* **.Scope**

Vlastnost ServiceInterval

Pouze pro čtení. Atribut MQI QServiceInterval .

Definováno v: Třída MQQueue

Datový typ: Long

Syntaxe: Pro získání: *serviceinterval & = MQQueue.* **ServiceInterval**

Vlastnost události ServiceInterval

Pouze pro čtení. Atribut události rozhraní MQI QServiceInterval.

Definováno v: Třída MQQueue

Datový typ: Long

Hodnoty:

- MQQSIE_HIGH
- MQQSIE_OK
- MQQSIE_NONE

Syntaxe: Chcete-li získat: *serviceintervalevent & = MQQueue.* **ServiceIntervalUdálost**

Vlastnost Shareability

Pouze pro čtení. Možnost sdílení fronty.

Definováno v: Třída MQQueue

Datový typ: Long

Hodnoty:

- MQQA_SHAREABLE
- MQQA_NOT_SHAREABLE

Syntaxe: Chcete-li získat: *shareability & = MQQueue* **.Shareability**

Vlastnost názvu TransmissionQueue

Pouze pro čtení. Název přenosové fronty. Platí pouze pro vzdálené fronty.

Definováno v: Třída MQQueue

Typ dat: Řetězec 48 znaků

Syntaxe: Chcete-li získat: *transqname $= MQQueue* **.TransmissionQueueName**

Vlastnost TriggerControl

Čtení a zápis. Řízení spouštěče.

Definováno v: Třída MQQueue

Datový typ: Long

Hodnoty:

- MQTC_OFF
- MQTC_ON

Syntaxe: Chcete-li získat následující informace: *trigcontrol & = MQQueue* **.TriggerControl**

Pro nastavení: *MQQueue* **.TriggerControl** = *trigcontrol &*

Vlastnost TriggerData

Čtení a zápis. Data spouątěče.

Definováno v: Třída MQQueue

Typ dat: Řetězec o délce 64 znaků

Syntaxe: Chcete-li získat: *trigdata $= MQQueue* **.TriggerData**

Chcete-li nastavit: *MQQueue* **.TriggerData** = *trigdata $*

Vlastnost TriggerDepth

Čtení a zápis. Počet zpráv, které musí být ve frontě, než je zapsána zpráva spouštěče.

Definováno v: Třída MQQueue

Datový typ: Long

Syntaxe: Chcete-li získat: *trigdepth & = MQQueue* **.TriggerDepth**

Pro nastavení: *MQQueue* **.TriggerDepth** = *trigdepth &*

Vlastnost priority TriggerMessage

Čtení a zápis. Priorita zprávy prahové hodnoty pro spouątěče.

Definováno v: Třída MQQueue

Datový typ: Long

Syntaxe: Chcete-li získat: *trigmesspriority & = MQQueue* **.TriggerMessagePriority**

Pro nastavení: *MQQueue* **.TriggerMessagePriority** = *trigmesspriority &*

Vlastnost TriggerType

Čtení a zápis. Typ spouątěče.

Definováno v: Třída MQQueue

Datový typ: Long

Hodnoty:

- MQTTE_NONE
- NEJPRVE MQTT_FIRST
- MQTT_EVERY
- MQTT_DEPTH

Syntaxe: Chcete-li získat: *trigtype & = MQQueue* **.TriggerType**

Chcete-li nastavit: *MQQueue* **.TriggerType** = *Trigtype &*

vlastnost použití

Pouze pro čtení. Označuje, pro kterou frontu se používá fronta.

Definováno v: Třída MQQueue

Datový typ: Long

Hodnoty:

- MQUS_NORMAL
- PŘENOS MQUS_TRANSMISSION

Syntaxe: Chcete-li získat: *usage & = MQQueue* **.Využití**

Metoda ClearErrorCodes

Resetuje CompletionCode na MQCC_OK a ReasonCode na MQRC_NONE jak pro třídu MQQueue, tak pro třídu MQSession.

Definováno v: Třída MQQueue

Syntaxe:

```
Call MQQueue.ClearErrorCodes()
```

Metoda Close

Zavře frontu s použitím aktuálních hodnot parametru CloseOptions.

Definováno v: Třída MQQueue

Syntaxe:

```
Call MQQueue.Close()
```

metoda GET

Načte zprávu z fronty.

Tato metoda vezme objekt MQMessage jako parametr s použitím některých polí v deskriptoru MQMD objektu jako vstupních parametrů. Konkrétně jsou použita pole MessageId a CorrelId , takže je důležité zajistit, aby byla tato pole nastavena podle potřeby. Další informace o těchto polích naleznete v části MsgId (MQBYTE24) a CorrelId (MQBYTE24).

Pokud metoda selže, objekt MQMessage se nezmění. Pokud uspěje, datové části MQMD a Message Data objektu MQMessage budou nahrazeny daty MQMD a Data zprávy z příchozí zprávy. Vlastnosti řízení MQMessage jsou nastaveny takto:

- **MessageLength** je nastavena na délku zprávy IBM MQ
- Hodnota **DataLength** je nastavena na délku zprávy IBM MQ .
- **DataOffset** je nastaven na nulu.

Definováno v:
Třída MQQueue

Syntaxe:

```
Call MQQueue.Get(Message, GetMsgOptions, GetMsgLength)
```

Parametry

Zpráva:

Objekt MQMessage Object reprezentující zprávu, která má být načtena.

Volby GetMsg:

Volitelný objekt Volby MQGetMessagepro řízení operace get. Není-li tento parametr zadán, použijí se výchozí volby MQGetMessage.

GetMsgDélka:

Volitelná hodnota délky 2 nebo 4, která určuje maximální délku zprávy IBM MQ , která je načtena z fronty.

Je-li zadána volba MQGMO_ACCEPT_TRUNCATED_MSG, operace GET se úspěšně provede s kódem dokončení MQCC_WARNING a kódem příčiny MQRC_TRUNCATED_MSG_ACCEPTED, pokud velikost zprávy překročí zadanou délku.

Položka MessageData uchovává první GetMsgLength bajtů dat.

Je-li zadána hodnota MQGMO_ACCEPT_TRUNCATED_MSG ***není*** a velikost zprávy překračuje uvedenou délku, je vrácen kód dokončení operace MQCC_FAILED spolu s kódem příčiny MQRC_TRUNCATED_MESSAGE_FAILED.

Není-li definován obsah vyrovnávací paměti zpráv, je celková délka zprávy nastavena na plnou délku zprávy, která by byla načtena.

Není-li parametr délky zprávy zadán, bude délka vyrovnávací paměti zpráv automaticky upravena alespoň na velikost příchozí zprávy.

Otevřít metodu

Otevře frontu s použitím aktuálních hodnot:

1. QueueName
2. QueueManagerName
3. AlternateUserid
4. Název DynamicQueue

Definováno v:
Třída MQQueue

Syntaxe:

```
Call MQQueue.Open()
```

metoda PUT

Umístí zprávu do fronty.

Tato metoda přijímá objekt MQMessage jako parametr. Vlastnosti objektu MQMD (Message Descriptor) tohoto objektu mohou být v důsledku této metody změněny. Hodnoty, které mají ihned po provedení této metody, jsou hodnoty, které byly vloženy do IBM MQ.

Úpravy objektu MQMessage po dokončení operace Put nemají vliv na skutečnou zprávu ve frontě produktu IBM MQ .

Definováno v:
Třída MQQueue

Syntaxe:

```
Call MQQueue.Put(Message, PutMsgOptions)
```

Parametry
Zpráva

Objekt MQMessage reprezentující zprávu, která má být vložena.

Volby PutMsg

Objekt voleb MQPutMessageobsahující volby pro řízení operace vložení. Nejsou-li tyto hodnoty zadány, použijí se výchozí volby PutMessage.

Třída MQMessage

Tato třída představuje zprávu IBM MQ . Obsahuje vlastnosti pro zapouzdření deskriptoru zpráv produktu IBM MQ (MQMD) a poskytuje vyrovnávací paměť pro uchovávání dat zpráv definovaných aplikací.

Třída obsahuje metody zápisu pro kopírování dat z aplikace ActiveX do objektu MQMessage. Podobně třída zahrnuje metody čtení pro kopírování dat z objektu MQMessage do aplikace ActiveX . Třída spravuje alokaci a dealokaci paměti pro vyrovnávací paměť automaticky. Aplikace nemusí deklarovat velikost

vyrovnávací paměti při vytvoření objektu MQMessage, protože vyrovnávací paměť dorůstá do přijetí dat, která jsou do ní zapsána.

Pokud velikost vyrovnávací paměti překročí vlastnost MaximumMessageLength dané fronty, nelze do fronty IBM MQ vložit zprávu.

Po vytvoření objektu MQMessage lze objekt MQMessage umístit do fronty produktu IBM MQ pomocí metody MQQueue.Put . Tato metoda vezme kopii dat MQMD a datových částí zprávy objektu a umístí tuto kopii do fronty. Aplikace proto může po vložení upravit nebo odstranit objekt MQMessage, aniž by došlo k ovlivnění zprávy ve frontě IBM MQ . Správce front může upravit některá pole v deskriptoru MQMD při kopírování zprávy do fronty produktu IBM MQ .

Příchozí zprávu lze číst do objektu MQMessage pomocí metody MQQueue.Get . Tím se nahradí veškerá data MQMD nebo zprávy, která mohla být v objektu MQMessage s hodnotami z příchozí zprávy, která již byla v objektu MQMessage. Upravuje velikost vyrovnávací paměti dat objektu MQMessage tak, aby odpovídala velikosti příchozích dat zprávy.

Obsažení

Zprávy jsou obsaženy ve třídě MQSession.

VYTVOŘENÍ

Volba **Nový** vytvoří objekt MQMessage. Jeho vlastnosti deskriptoru zpráv jsou na počátku nastaveny na výchozí hodnoty a její vyrovnávací paměť dat je prázdná.

Syntaxe

```
Dim msg As New MQMessage
```

, nebo

```
Set msg = New MQMessage
```

Vlastnosti

Vlastnosti ovládacích prvků jsou:

- “Vlastnost CompletionCode” na stránce 601
- “Vlastnost DataLength” na stránce 602
- “Vlastnost DataOffset” na stránce 602
- “Vlastnost MessageLength” na stránce 602
- “Vlastnost ReasonCode” na stránce 603
- “Vlastnost ReasonName” na stránce 603

Vlastnosti deskriptoru zpráv jsou:

- “Vlastnost AccountingToken” na stránce 603
- “AccountingTokenHexadecimální vlastnost” na stránce 603
- “Datová vlastnost ApplicationId” na stránce 603
- “vlastnost dat ApplicationOrigin” na stránce 604
- “Vlastnost BackoutCount” na stránce 604
- “Vlastnost CharacterSet” na stránce 604
- “Vlastnost CorrelationId” na stránce 605
- “CorrelationIdHexadecimální vlastnost” na stránce 605

Metody

- “Metoda ReadUnsignedByte” na stránce 615
- “Metoda ReadUTF” na stránce 615
- “Metoda ResizeBuffer” na stránce 616
- “Metoda zápisu” na stránce 616
- “Metoda WriteBoolean” na stránce 616
- “Metoda WriteByte” na stránce 617
- “Metoda WriteDecimal2” na stránce 617
- “Metoda WriteDecimal4” na stránce 617
- “Metoda WriteDouble” na stránce 617
- “Metoda WriteDouble4” na stránce 618
- “Metoda WriteFloat” na stránce 618
- “Metoda WriteInt2” na stránce 618
- “Metoda WriteInt4” na stránce 618
- “Metoda WriteLong” na stránce 619
- “Metoda WriteNullTerminatedString” na stránce 619
- “Metoda WriteShort” na stránce 619
- “Metoda WriteString” na stránce 620
- “Metoda WriteUInt2” na stránce 620
- “WriteUnsignedBytová metoda” na stránce 620
- “Metoda WriteUTF” na stránce 620

Přístup k vlastnosti

Všechny vlastnosti lze číst kdykoli.

Řídicí vlastnosti jsou určeny pouze pro čtení, kromě DataOffset , který je čtení-zápis. Vlastnosti deskriptoru zpráv jsou všechny pro čtení-zápis, kromě BackoutCount a TotalMessageLength, které jsou určeny pouze pro čtení.

Mějte však na paměti, že některé vlastnosti MQMD mohou správce front upravit při vložení zprávy do fronty produktu IBM MQ . Podrobné informace o tom, jak mohou být upraveny, najdete v polích v MQMD .

Převod dat

Můžete předat binární data do zprávy IBM MQ nastavením vlastnosti **`CharacterSet`** tak, aby odpovídala identifikátoru kódované znakové sady správce front (MQCCSI_Q_MMGR), a předáváním dat do zprávy jako řetězec. Pokud řetězec potřebuje zahrnout kód Unicode nebo ASCII, můžete použít funkci chr $, abyste je převedli na řetězcový formát.

Metody čtení a zápisu provádějí převod dat. Konvertují mezi vnitřním formátorem ActiveX a formáty zpráv produktu IBM MQ definované vlastnostmi Encoding a CharacterSet z deskriptoru zpráv. Při zápisu zprávy nastavte hodnoty do kódování a CharacterSet , která odpovídá charakteristikám příjemce zprávy před vydáním metody Write. Při čtení zprávy se tento krok obvykle nevyžaduje, protože tyto hodnoty budou nastaveny z hodnot v příchozím MQMD.

Jedná se o další krok konverze dat, který se stane po provedení jakékoli konverze provedené metodou MQQueue.Get .

Vlastnost CompletionCode

Pouze pro čtení. Vrací kód dokončení IBM MQ nastavený nejnovější metodou nebo úrovní přístupu k vlastnosti vydaným pro tento objekt.

Definováno v: třídě MQMessage

Datový typ: Long

Hodnoty:

- MQCC_OK
- VAROVÁNÍ MQCC_WARNING
- SELHÁNÍ MQCC_FAILED

Syntaxe: Chcete-li získat: *completioncode & = MQMessage* **.CompletionCode**

Vlastnost DataLength

Pouze pro čtení. Tato vlastnost vrací hodnotu:

```
MQMessage.MessageLength - MQMessage.DataOffset
```

Lze jej použít před metodou čtení, abyste zkontrolli, že očekávaný počet znaků je ve skutečnosti přítomen ve vyrovnávací paměti.

Počáteční hodnota je nula.

Definováno v: třídě MQMessage

Datový typ: Long

Syntaxe: Chcete-li získat: *bytesaleft & = MQMessage* **.DataLength**

Vlastnost DataOffset

Čtení a zápis. Aktuální pozice v části Data zprávy objektu zprávy.

Hodnota je vyjádřena jako bajtový posun od začátku vyrovnávací paměti dat zprávy; první znak ve vyrovnávací paměti odpovídá hodnotě DataOffset nulové hodnoty.

Metoda čtení nebo zápisu zahajuje svou činnost na znaku, na který se odkazuje DataOffset. Tyto metody zpracují data ve vyrovnávací paměti sekvenčně z této pozice a aktualizují hodnotu DataOffset tak, aby ukazovala na bajt (je-li nějaký) bezprostředně za posledním zpracovávanými bajty.

Hodnota DataOffset může obsahovat pouze hodnoty v rozsahu od nuly do MessageLength včetně. Když DataOffset = MessageLength ukazuje na konec, to je první neplatný znak vyrovnávací paměti. Metody zápisu jsou v této situaci povoleny-rozšiřují data ve vyrovnávací paměti a zvyšují MessageLength o počtu přidaných bajtů. Čtení nad konec vyrovnávací paměti není platné.

Počáteční hodnota je nula.

Definováno v: třídě MQMessage

Datový typ: Long

Syntaxe: Chcete-li získat: *curpos & = MQMessage* **.DataOffset**

Pro nastavení: *MQMessage* **.DataOffset** = *curpos &*

Vlastnost MessageLength

Pouze pro čtení. Vrátí celkovou délku části dat zprávy objektu zprávy ve znacích bez ohledu na hodnotu parametru DataOffset.

Počáteční hodnota je nula. Je nastavena na příchozí délku zprávy po vyvolání metody Get, která odkazuje na tento objekt zprávy. Tato hodnota se zvýší, pokud aplikace používá metodu Write k přidání dat do objektu. To není ovlivněno metodami čtení.

Definováno v: třídě MQMessage

Datový typ: Long

Syntaxe: Chcete-li získat: *msglolength & = MQMessage* **.MessageLength**

Vlastnost ReasonCode

Pouze pro čtení. Vrátí kód příčiny nastavený nejnovější metodou nebo úrovní přístupu k vlastnosti vydaným pro tento objekt.

Definováno v: třídě MQMessage

Datový typ: Long

Hodnoty:

- Viz termín completion API a kódy příčin.

Syntaxe: Chcete-li získat: *reasoncode & = MQMessage* **.ReasonCode**

Vlastnost ReasonName

Pouze pro čtení. Vrátí symbolický název posledního kódu příčiny. Příklad: "MQRC_QMGR_NOT_AVAILABLE". **Definováno v:** třídě MQMessage

Datový typ: Řetězec

Hodnoty:

- Viz termín completion API a kódy příčin.

Syntaxe: Chcete-li získat: *reasonname $= MQMessage* **.ReasonName**

Vlastnost AccountingToken

Čtení a zápis. MQMD AccountingToken -část kontextu identity zprávy.

Jeho počáteční hodnota je všechny hodnoty null.

Definováno v: třídě MQMessage

Typ dat: Řetězec o délce 32 znaků

Syntaxe: Chcete-li získat: *actoken $= MQMessage* **.AccountingToken**

Chcete-li nastavit: *MQMessage.* **AccountingToken** = *actoken $*

Další informace o tom, kdy musíte použít AccountingTokenHex na místě vlastnosti AccountingToken , naleznete v tématu “Vlastnosti deskriptoru zpráv” na stránce 563 .

AccountingTokenHexadecimální vlastnost

Čtení a zápis. MQMD AccountingToken -část kontextu identity zprávy.

Každé dva znaky reprezentují hexadecimální ekvivalent jednoho znaku ASCII. Například dvojice znaků "6" a "1" představují jediný znak "A", dvojice znaků "6" a "2" představují jediný znak "B" atd.

Je třeba zadat 64 platných hexadecimálních znaků.

Jeho počáteční hodnota je "0 ... 0"

Definováno v: třídě MQMessage

Typ dat: Řetězec 64 hexadecimálních znaků představujících 32 znaků ASCII

Syntaxe: Chcete-li získat: *actokenh $= MQMessage* **.AccountingTokenHex**

Chcete-li nastavit: *MQMessage.* **AccountingTokenHex** = *actokenh $*

Další informace o tom, kdy musíte použít AccountingTokenHex na místě vlastnosti AccountingToken , naleznete v tématu “Vlastnosti deskriptoru zpráv” na stránce 563 .

Datová vlastnost ApplicationId

Čtení a zápis. Data produktu MQMD ApplIdentityData-Část kontextu identity zprávy.

Jeho počáteční hodnota je prázdná.

Definováno v: třídě MQMessage

Typ dat: Řetězec o délce 32 znaků

Syntaxe: Chcete-li získat: *applid $= MQMessage* **.ApplicationIdData**

Nastavení: *MQMessage* **.ApplicationIdData** = *applid $*

vlastnost dat ApplicationOrigin

Čtení a zápis. MQMD ApplOriginData-Část kontextu zprávy o původu zprávy.

Jeho počáteční hodnota je prázdná.

Definováno v: třídě MQMessage

Typ dat: Řetězec o délce 4 znaků

Syntaxe: Chcete-li získat: *applor $= MQMessage* **.ApplicationOriginData**

Chcete-li nastavit: *MQMessage* **.ApplicationOriginData** = *applor $*

Vlastnost BackoutCount

Pouze pro čtení. MQMD BackoutCount.

Jeho počáteční hodnota je 0

Definováno v: třídě MQMessage

Datový typ: Long

Syntaxe: Chcete-li získat: *backoutct & = MQMessage* **.BackoutCount**

Vlastnost CharacterSet

Čtení a zápis. MQMD CodedCharSetId.

Jeho počáteční hodnota je speciální hodnota **MQCCSI_Q_MGR**.

Je-li vlastnost CharacterSet nastavena na hodnotu **MQCCSI_Q_MGR**, použije se kódová stránka pro aktuální národní prostředí pro znakovou konverzi v metodě WriteString . Pro serverové aplikace je použitá kódová stránka kódovou stránkou správce front. Pro klientské aplikace se jedná o výchozí aktuální kódovou stránku národního prostředí.

Příklad:

```
msg.CharacterSet = MQCCSI_Q_MGR
msg.WriteString(chr$(n))
```

kde 'n' je větší nebo rovno nule a menší nebo rovno 255, vede k tomu, že jediný bajt hodnoty' n' je zapsán do vyrovnávací paměti.

Definováno v: třídě MQMessage

Datový typ: Long

Syntaxe: Chcete-li získat: *:30ccid& = MQMessage* **.CharacterSet**

Pro nastavení: *MQMessage* **.CharacterSet** = *ccid &*

Příklad

Pokud chcete řetězec zapsaný v kódové stránce 437, zadejte:

```
Message.CharacterSet = 437
Message.WriteString ("string to be written")
```

Před zadáním jakéhokoli volání WriteString nastavte hodnotu, kterou chcete v souboru CharacterSet .

Vlastnost CorrelationId

Čtení a zápis. Identifikátor CorrelationId , který má být zahrnut do MQMD zprávy při vložení do fronty. Také ID, které se bude porovnávat při získávání zprávy z fronty.

Jeho počáteční hodnota je null.

Definováno v: třídě MQMessage

Typ dat: Řetězec o délce 24 znaků

Syntaxe: Chcete-li získat: *correlid $= MQMessage* **.CorrelationId** To set: *MQMessage* **.CorrelationId** = *correlid $*

Další informace o tom, kdy musíte použít CorrelationIdHex místo vlastnosti CorrelationId , najdete v tématu “Vlastnosti deskriptoru zpráv” na stránce 563 .

CorrelationIdHexadecimální vlastnost

Čtení a zápis. Identifikátor CorrelationId , který má být zahrnut do MQMD zprávy při vložení do fronty. Také CorrelationId , které se má porovnávat při získávání zprávy z fronty.

Každé dva znaky řetězce reprezentují hexadecimální ekvivalent jednoho znaku ASCII. Například dvojice znaků "6" a "1" představují jediný znak "A", dvojice znaků "6" a "2" představují jediný znak "B" atd.

Je třeba zadat 48 platných hexadecimálních znaků.

Jeho počáteční hodnota je "0 ... 0".

Definováno v: třídě MQMessage

Typ dat: Řetězec 48 hexadecimálních znaků představujících 24 znaků ASCII

Syntaxe: Chcete-li získat: *correlidh $= MQMessage* **.CorrelationIdHex**

Nastavení: *MQMessage* **.CorrelationIdHex** = *correlidh $*

Informace o tom, kdy musíte použít CorrelationIdHex místo vlastnosti CorrelationId , najdete v části “Vlastnosti deskriptoru zpráv” na stránce 563 .

Vlastnost kódování

Čtení a zápis. Pole MQMD, které identifikuje znázornění použité pro číselné hodnoty v datech zprávy aplikace.

Jeho počáteční hodnota je speciální hodnota MQENC_NATIVE, která se liší podle platformy.

Tato vlastnost je používána následujícími metodami:

- Metoda ReadDecimal2
- Metoda ReadDecimal4
- Metoda ReadDouble
- Metoda ReadDouble4
- Metoda ReadFloat
- Metoda ReadInt2
- Metoda ReadInt4
- Metoda ReadLong
- Metoda ReadShort
- Metoda ReadUInt2
- Metoda WriteDecimal2
- Metoda WriteDecimal4
- Metoda WriteDouble

- Metoda WriteDouble4
- Metoda WriteFloat
- Metoda WriteInt2
- Metoda WriteInt4
- Metoda WriteLong
- Metoda WriteShort
- Metoda WriteUInt2

Definováno v: třídě MQMessage

Datový typ: Long

Syntaxe: Chcete-li získat: *encoding & = MQMessage* **.Kódování** Chcete-li nastavit: *MQMessage* **.Kódování** = *encoding &*

Pokud připravujete zápis dat do vyrovnávací paměti zpráv, měli byste toto pole nastavit tak, aby odpovídalo charakteristikám přijímající platformy správce front, pokud přijímající správce front není schopen provést vlastní převod dat.

Vlastnost vypršení platnosti

Čtení a zápis. Pole vypršení platnosti MQMD se očekává v desetinách sekundy.

Jeho počáteční hodnota je speciální hodnota MQEI_UNLIMITED.

Definováno v: třídě MQMessage

Datový typ: Long

Syntaxe: Chcete-li získat: *expiry & = MQMessage* **.Vypršení platnosti**

Nastavení: *MQMessage* **.Expiry** = *expiry &*

Vlastnost Feedback

Čtení a zápis. Pole zpětné vazby MQMD.

Jeho počáteční hodnota je speciální hodnota MQFB_NONE.

Definováno v: třídě MQMessage

Datový typ: Long

Hodnoty:

- Viz Váš názor.

Syntaxe: Chcete-li získat: *feedback & = MQMessage* **.Váš názor**

Nastavení: *MQMessage* **.Zpětná vazba** = *feedback &*

Vlastnost formátu

Čtení a zápis. Pole formátu MQMD. Poskytuje název vestavěného nebo uživatelem definovaného formátu, který popisuje povahu dat zprávy.

Jeho počáteční hodnotou je speciální hodnota MQFMT_NONE.

Definováno v: třídě MQMessage

Typ dat: Řetězec o délce 8 znaků

Syntaxe: Chcete-li získat: *format $= MQMessage* **.Formát**

Nastavení: *MQMessage* **.Formát** = *formát $*

Vlastnost GroupId

Čtení a zápis. Hodnota GroupId , která má být zahrnuta do zprávy MQPMR zprávy při vložení do fronty. Také ID, které se bude porovnávat při získávání zprávy z fronty. Jeho počáteční hodnota je všechny hodnoty null.

Definováno v:
Třída MQMessage

Datový typ:
Řetězec o délce 24 znaků

Syntaxe: Chcete-li získat: *groupid $= MQMessage*. **GroupId**

Nastavení: *MQMessage*. **GroupId** = *groupid $*

Další informace o tom, kdy je třeba použít parametr GroupIdHex místo vlastnosti GroupId , naleznete v tématu "Vlastnosti deskriptoru zpráv" na stránce 563 .

GroupIdHexadecimální vlastnost

Čtení a zápis. Hodnota GroupId , která má být zahrnuta do zprávy MQPMR zprávy při vložení do fronty. Také ID, které se bude porovnávat při získávání zprávy z fronty.

Každé dva znaky řetězce reprezentují hexadecimální ekvivalent jednoho znaku ASCII. Například dvojice znaků "6" a "1" představují jediný znak "A", dvojice znaků "6" a "2" představují jediný znak "B" a tak dále.

Je třeba zadat 48 platných hexadecimálních znaků.

Jeho počáteční hodnota je "0 ... 0".

Definováno v:
Třída MQMessage

Datový typ:
Řetězec 48 hexadecimálních znaků představujících 24 znaků ASCII.

Syntaxe: Chcete-li získat: *groupidh $= MQMessage*. **GroupIdHex**

Nastavení: *MQMessage*. **GroupIdHex** = *groupidh $*

Další informace o tom, kdy je třeba použít parametr GroupIdHex místo vlastnosti GroupId , naleznete v tématu "Vlastnosti deskriptoru zpráv" na stránce 563 .

Vlastnost MessageData

Čtení a zápis. Načte nebo nastaví celý obsah zprávy jako znakový řetězec.

Definováno v: třídě MQMessage

Datový typ: Varianta

Poznámka: Datový typ používaný touto vlastností je Variant, ale MQAX očekává, že se jedná o typ varianty String. Pokud předáte v jiné variantě než tento typ, bude vrácena chyba MQRC_OBJECT_TYPE_ERROR.

Syntaxe: Chcete-li získat: *String$* = *MQMessage* **.MessageData**

Pro nastavení: *MQMessage* **.MessageData** = *String$*

Vlastnost MessageFlags

Čtení a zápis. Příznaky zprávy určující řídicí informace Segmentace. Počáteční hodnota je 0.

Definováno v:
Třída MQMessage

Datový typ:
Dlouhý

Hodnoty:
Viz MsgFlags (MQLONG).

Syntaxe: Chcete-li získat: *messageflags & = MQMessage*. **MessageFlags**

Nastavení: *MQMessage*. **MessageFlags** = *messageflags &*

Vlastnost MessageId

Čtení a zápis. Hodnota MessageId , která má být zahrnuta do MQMD zprávy při vložení do fronty. Také ID, které se bude porovnávat při získávání zprávy z fronty.

Jeho počáteční hodnota je všechny hodnoty null.

Definováno v: třídě MQMessage

Typ dat: Řetězec o délce 24 znaků

Syntaxe: Chcete-li získat: *messageid $= MQMessage* **.MessageId**

Pro nastavení: *MQMessage* **.MessageId** = *messageid $*

Další informace o tom, kdy je třeba použít MessageIdHex místo vlastnosti MessageId , naleznete v tématu “Vlastnosti deskriptoru zpráv” na stránce 563 .

MessageIdHexadecimální vlastnost

Čtení a zápis. Hodnota MessageId , která má být zahrnuta do MQMD zprávy při vložení do fronty. Také MessageId , které má být porovnávána při získávání zprávy z fronty.

Každé dva znaky řetězce reprezentují hexadecimální ekvivalent jednoho znaku ASCII. Například dvojice znaků "6" a "1" představují jediný znak "A", dvojice znaků "6" a "2" představují jediný znak "B" atd.

Je třeba zadat 48 platných hexadecimálních znaků.

Jeho počáteční hodnota je "0 ... 0".

Definováno v: třídě MQMessage

Typ dat: Řetězec 48 hexadecimálních znaků představujících 24 znaků ASCII

Syntaxe: Chcete-li získat: *messagesh $= MQMessage* **.MessageIdHex**

Nastavení: *MQMessage* **.MessageIdHex** = *messageidh $* .

Další informace o tom, kdy je třeba použít MessageIdHex místo vlastnosti MessageId , naleznete v tématu “Vlastnosti deskriptoru zpráv” na stránce 563 .

Vlastnost čísla MessageSequence

Čtení a zápis. Informace o posloupnosti identifikující zprávu v rámci skupiny. Počáteční hodnota je 1.

Definováno v:
Třída MQMessage

Datový typ:
Dlouhý

Hodnoty:
Viz MsgSeqNumber (MQLONG).

Syntaxe: Chcete-li získat: *sequencenumber & = MQMessage*. **SequenceNumber**

Nastavení: *MQMessage*. **SequenceNumber** = *sequenciumber &*

Vlastnost MessageType

Čtení a zápis. Pole MQMD MsgType .

Jeho počáteční hodnota je MQMT_DATAGRAM.

Definováno v: třídě MQMessage

Datový typ: Long

Hodnoty:

- Viz MsgType (MQLONG).

Syntaxe: Chcete-li získat: *msgtype & = MQMessage* **.MessageType**

Pro nastavení: *MQMessage* **.MessageType** = *msgtype &*

Vlastnost Posunutí

Čtení a zápis. Posunutí v segmentované zprávě. Počáteční hodnota je 0.

Definováno v:
Třída MQMessage

Datový typ:
Dlouhý

Hodnoty:
Viz Posunutí (MQLONG).

Syntaxe: Chcete-li získat: *offset & = MQMessage*. **Offset**

Nastavení: *MQMessage*. **Posunutí** = *posun &*

Vlastnost OriginalLength

Čtení a zápis. Původní délka segmentované zprávy. Počáteční hodnota je MQOL_UNDEFINED.

Definováno v:
Třída MQMessage

Datový typ:
Dlouhý

Hodnoty:
Viz OriginalLength (MQLONG).

Syntaxe: Chcete-li získat: *originallength & = MQMessage*. **OriginalLength**

Nastavení: *MQMessage*. **OriginalLength** = *původní_délka &*

Vlastnost Persistence

Čtení a zápis. Nastavení perzistence zprávy.

Jeho počáteční hodnota je MQPER_PERSISTENCE_AS_Q_DEF.

Definováno v: třídě MQMessage

Datový typ: Long

Syntaxe: Chcete-li získat: *persist & = MQMessage* **.Perzistence**

Nastavení: *MQMessage* **.Persistence** = *persist &*

Vlastnost priority

Čtení a zápis. Priorita zprávy.

Jeho počáteční hodnota je speciální hodnota MQPRI_PRIORITY_AS_Q_DEF

Definováno v: třídě MQMessage

Datový typ: Long

Syntaxe: Chcete-li získat: *priority & = MQMessage* **.Priorita.**

Nastavení: *MQMessage* **.Priorita** = *priorita &*

Vlastnost názvu PutApplication

Čtení a zápis. Název MQMD PutAppl-část kontextu Původ zprávy.

Jeho počáteční hodnota je prázdná.

Definováno v: třídě MQMessage

Typ dat: Řetězec obsahující 28 znaků

Syntaxe: Chcete-li získat: *putapplnm $= MQMessage* **.PutApplicationName**

Nastavení: *MQMessage* **.PutApplicationName** = *putapplnm $*

Vlastnost typu PutApplication

Čtení a zápis. Typ záhlaví MQMD PutAppl-část kontextu Původ zprávy.

Jeho počáteční hodnota je MQAT_NO_CONTEXT

Definováno v: třídě MQMessage

Datový typ: Long

Hodnoty:

- Viz termín PutApplType (MQLONG).

Syntaxe: Chcete-li získat: *putappltp & = MQMessage* **.PutApplicationType**

Nastavení: *MQMessage* **.PutApplicationType** = *putappltp &*

Vlastnost Time PutDate

Čtení/zápis. Tato vlastnost kombinuje pole MQMD PutDate a PutTime . Jedná se o části kontextu Původ zprávy, které označují, kdy byla zpráva vložena.

Rozšíření ActiveX se převádí mezi formátem data a času ActiveX a formáty data a času použité v produktu IBM MQ MQMD. Je-li přijata zpráva, která má neplatnou hodnotu PutDate nebo PutTime, pak je vlastnost PutDateTime po metodě get nastavena na EMPTY.

Jeho počáteční hodnota je EMPTY.

Definováno v: třídě MQMessage

Typ dat: Varianta typu 7 (datum/čas) nebo EMPTY.

Syntaxe: Chcete-li získat: *datetime = MQMessage* **.PutDateTime**

Pro nastavení: *MQMessage* **.PutDateTime** = *datetime* .

Vlastnost ReplyToQueueManagerName

Čtení a zápis. Pole QMgr ReplyToMQMD.

Jeho počáteční hodnota je prázdná

Definováno v: třídě MQMessage

Typ dat: Řetězec 48 znaků

Syntaxe: Chcete-li získat následující informace: *replytoqmgr $= MQMessage* **.ReplyToQueueManagerName**

Pro nastavení: *MQMessage* **.ReplyToQueueManagerName** = *replytoqmgr $*

Vlastnost ReplyToQueueName

Čtení a zápis. Pole Q ReplyToMQMD.

Jeho počáteční hodnota je prázdná

Definováno v: třídě MQMessage

Typ dat: Řetězec 48 znaků

Syntaxe: Chcete-li získat následující informace: *replytoq $= MQMessage* **.ReplyToQueueName**

Pro nastavení: *MQMessage* **.ReplyToQueueName** = *replytoq $*

Vlastnost sestavy

Čtení a zápis. Volby sestavy zprávy.

Jeho počáteční hodnota je MQRO_NONE.

Definováno v: třídě MQMessage

Datový typ: Long

Hodnoty:

- Viz Sestava.

Syntaxe: Chcete-li získat: *report & = MQMessage* **.Sestava**

Nastavení: *MQMessage* **.Sestava** = *report &*

Vlastnost TotalMessageLength

Pouze pro čtení. Načte délku poslední zprávy přijaté příkazem MQGET. Pokud nebyla zpráva zkrácena, tato hodnota se rovná hodnotě vlastnosti MessageLength .

Definováno v: třídě MQMessage

Datový typ: Long

Syntaxe: Chcete-li získat: *totalmessagelength & = MQMessage* **.TotalMessageLength**

vlastnost UserId

Čtení a zápis. MQMD UserIdentifier -část kontextu identity zprávy.

Jeho počáteční hodnota je prázdná.

Definováno v: třídě MQMessage

Datový typ: Řetězec o délce 12 znaků

Syntaxe: Chcete-li získat: *userid $= MQMessage* **.UserId**

Pro nastavení: *MQMessage* **.UserId** = *userid $*

Metoda ClearErrorCodes

Resetuje CompletionCode na MQCC_OK a ReasonCode na MQRC_NONE jak pro třídu MQMessage, tak pro třídu MQSession.

Definováno v: třídě MQMessage

Syntaxe:

```
Call MQMessage.ClearErrorCodes()
```

Metoda ClearMessage

Tato metoda vymaže část vyrovnávací paměti dat objektu MQMessage. Veškerá data zprávy v datové vyrovnávací paměti se ztratí, protože MessageLength, DataLengtha DataOffset jsou všechny nastaveny na nulu.

Část MQMD (Message Descriptor) není ovlivněna; aplikace může vyžadovat úpravu některých z polí MQMD předtím, než bude znovu použit objekt MQMessage. Chcete-li nastavit pole MQMD zpět, použijte volbu Nový pro nahrazení objektu novou instancí.

Definováno v: třídě MQMessage

Syntaxe:

```
Call MQMessage.ClearMessage()
```

Metoda čtení

Čte posloupnost bajtů z vyrovnávací paměti zpráv do bajtového pole. Hodnota DataOffset se inkrementuje a sníží se o délku dat o počet přečtených bajtů.

Definováno v:
Třída MQMessage

Syntaxe: Data = MQMessage. **Čtení** (len &)

Parametry:
len &: Long. Délka dat v bajtech, která se mají přečíst.

Metoda ReadBoolean

Čte 1bajtovou logickou hodnotu z aktuální pozice v vyrovnávací paměti zpráv a vrací 2bajtovou logickou hodnotu TRUE (-1) /FALSE (0). Hodnota DataOffset je zvýšena o jedničku a délka dat se sníží o jednu.

Definováno v:
Třída MQMessage

Syntaxe: *value = MQMessage*. **ReadBoolean**

Metoda ReadByte

Počínaje bajtem, na který se odkazuje DataOffset, čte metoda ReadByte 1 bajt z vyrovnávací paměti dat zprávy a vrací ji jako celočíselnou hodnotu typu Integer (podepsaná 2-bajt) v rozsahu -128 až 127.

Metoda selže, pokud je MQMessage.DataLength menší než 1, když je vydána.

Hodnota DataOffset se zvýší o 1 a hodnota DataLength se sníží o 1, je-li metoda úspěšná.

Bytem dat zprávy se předpokládá, že se jedná o podepsané binární celé číslo.

Definováno v:
Třída MQMessage

Syntaxe:
integerv% = MQMessage **.ReadByte**

Metoda ReadDecimal2

Přečte 2bajtové pakované desetinné číslo a vrátí jej jako podepsanou 2bajtovou celočíselnou hodnotu. Hodnota DataOffset je zvýšena o dvě a délka dat se sníží o dvě.

Definováno v:
Třída MQMessage

Syntaxe: *value% = MQMessage*. **ReadDecimal2**

Metoda ReadDecimal4

Čte 4bajtové pakované desetinné číslo a vrátí ji jako 4bajtovou celočíselnou hodnotu se znaménkem. Hodnota DataOffset se zvýší o čtyři a Data Length se sníží o čtyři.

Definováno v:
Třída MQMessage

Syntaxe:

```
Call value& = MQMessage.ReadDecimal4
```

Metoda ReadDouble

Počínaje bajtem, na který se odkazuje DataOffset, metoda ReadDouble čte 8 bajtů z vyrovnávací paměti dat zprávy a vrací je jako hodnotu s plovoucí řádovou čárkou Double (podepsaná 8bajtová).

Metoda selže, pokud je MQMessage.DataLength menší než 8, když je vydána.

Hodnota DataOffset se zvýší o 8 a hodnota DataLength se sníží o 8, je-li metoda úspěšná.

Předpokládá se, že 8 znaků dat zprávy je binární číslo s pohyblivou řádovou čárkou. Kódování je určeno vlastností MQMessage.Encoding . Všimněte si, že převod ze formátu System/360 není podporován.

Definováno v:
Třída MQMessage

Syntaxe:
doublev# = MQMessage **.ReadDouble**

Metoda ReadDouble4

Metody ReadDouble4 a WriteDouble4 jsou alternativy k ReadFloat a WriteFloat. Důvodem je to, že podporují 4bajtové hodnoty zpráv s pohyblivou řádovou čárkou System/390 , které jsou příliš velké pro převod na 4bajtový formát IEEE s plovoucí řádovou čárkou.

Počínaje bajtem označeným jako DataOffsetčte metoda ReadDouble4 4 bajty z vyrovnávací paměti dat zprávy a vrací je jako hodnotu s plovoucí řádovou čárkou (8bajtová).

Metoda selže, pokud je MQMessage.DataLength menší než 4, když je vydána.

Hodnota DataOffset se zvýší o 4 a hodnota DataLength se sníží o 4, je-li metoda úspěšná.

Předpokládá se, že 4 znaky dat zprávy jsou binární číslo s pohyblivou řádovou čárkou. Kódování je určeno vlastností MQMessage.Encoding . Všimněte si, že převod ze formátu System/360 není podporován.

Definováno v:
Třída MQMessage

Syntaxe:
doublev# = MQMessage. **ReadDouble4**

Metoda ReadFloat

Počínaje bajtem, na který odkazuje hodnota DataOffset, čte metoda ReadFloat 4 bajty z vyrovnávací paměti dat zprávy a vrací je jako hodnotu s plovoucí řádovou čárkou Single (podepsaná 4 bajty).

Metoda selže, pokud je MQMessage.DataLength menší než 4, když je vydána.

Hodnota DataOffset se zvýší o 4 a hodnota DataLength se sníží o 4, je-li metoda úspěšná.

Předpokládá se, že 4 znaky dat zprávy jsou číslo s pohyblivou řádovou čárkou. Kódování je určeno vlastností MQMessage.Encoding . Všimněte si, že převod ze formátu System/360 není podporován.

Definováno v:
Třída MQMessage

Syntaxe:
singlev! = MQMessage **.ReadFloat**

Metoda ReadInt2

Metoda je identická s metodou ReadShort .

Syntaxe:
intedgerv% = MQMessage **.ReadInt2**

Metoda ReadInt4

Tato metoda je identická s metodou ReadLong .

Syntaxe:
bigint & = MQMessage **.ReadInt4**

Metoda ReadLong

Počínaje bajtem, na který se odkazuje DataOffset, metoda ReadLong čte 4 bajty z vyrovnávací paměti dat zprávy a vrací je jako hodnotu typu Long (se znaménkem 4-byte).

Metoda selže, pokud je MQMessage.DataLength menší než 4, když je vydána.

Hodnota DataOffset se zvýší o 4 a hodnota DataLength se sníží o 4, je-li metoda úspěšná.

Předpokládá se, že 4 znaky dat zprávy jsou binární celé číslo. Kódování je určeno vlastností MQMessage.Encoding .

Definováno v:
Třída MQMessage

Syntaxe:
bigint & = MQMessage **.ReadLong**

Metoda ReadNullTerminatedString

Tato metoda je určena pro použití v místě ReadString , pokud řetězec může obsahovat vložené znaky null.

Tato metoda přečte uvedený počet bajtů z vyrovnávací paměti dat zprávy začínající na bajt, na který odkazuje hodnota DataOffset a vrací jej jako řetězec ActiveX . Pokud řetězec obsahuje vložený znak null před koncem, pak je délka vráceného řetězce redukována tak, aby odrážela pouze ty znaky před hodnotou null.

Hodnota DataOffset je inkrementována a hodnota DataLength se sníží o zadanou hodnotu bez ohledu na to, zda řetězec obsahuje vložené nulové znaky.

Předpokládá se, že znaky v datech zprávy jsou řetězcem v kódové stránce určené vlastností MQMessage.CharacterSet . Konverze na znázornění ActiveX se provádí pro aplikaci.

Definováno v:
Třída MQMessage

Syntaxe: *string $ = MQMessage.* **ReadNullTerminatedString(délka &)**

Parametry:
délka & *Long.* Délka pole řetězce v bajtech.

Metoda ReadShort

Počínaje bajtem, na který se odkazuje DataOffset, metoda ReadShort čte 2 bajty z vyrovnávací paměti dat zprávy a vrací je jako hodnotu typu Integer (podepsaná 2-byte).

Metoda selže, pokud je MQMessage.DataLength menší než 2, když je vydána.

Hodnota DataOffset se zvýší o 2 a hodnota DataLength se sníží o 2, pokud je metoda úspěšná.

Předpokládá se, že 2 znaky dat zprávy jsou binární celé číslo. Kódování je určeno vlastností MQMessage.Encoding .

Definováno v:
Třída MQMessage

Syntaxe:
integerv% = MQMessage **.ReadShort**

Metoda ReadString

Tato metoda čte n bajtů z vyrovnávací paměti dat zprávy začínající na bajt, na který odkazuje DataOffset a vrací jej jako řetězec ActiveX .

Metoda selže, pokud je MQMessage.DataLength menší než n, když je vydán.

Hodnota DataOffset je zvýšena o n a hodnota DataLength se sníží o n, pokud metoda uspěje.

Předpokládá se, že n znaků dat zprávy je řetězec v kódové stránce určené vlastností MQMessage.CharacterSet . Konverze na znázornění ActiveX se provádí pro aplikaci.

Definováno v: třídě MQMessage

Syntaxe: *stringv $= MQMessage* **.ReadString** *(délka &)*

Parametr
délka & Long. Délka pole řetězce v bajtech.

Metoda ReadUInt2

Počínaje bajtem, na který se odkazuje DataOffset, čte metoda ReadUInt2 2 bajty z vyrovnávací paměti dat zprávy a vrací je jako hodnotu typu Long (podepsaná 4-byte).

Metoda selže, pokud je MQMessage.DataLength menší než 2, když je vydána.

Hodnota DataOffset se zvýší o 2 a hodnota DataLength se sníží o 2, pokud je metoda úspěšná.

Předpokládá se, že 2 bajty dat zprávy jsou binární celé číslo bez znaménka. Kódování je určeno vlastností MQMessage.Encoding .

Definováno v:
Třída MQMessage

Syntaxe:
bigint & = MQMessage **.ReadUInt2**

Metoda ReadUnsignedByte

Počínaje bajtem, na který se odkazuje DataOffset, bajtová metoda ReadUnsignedčte 1 bajt z vyrovnávací paměti dat zprávy a vrací ji jako celočíselnou hodnotu typu Integer (podepsaná 2-bajt) v rozsahu od 0 do 255.

Metoda selže, pokud je MQMessage.DataLength menší než 1, když je vydána.

Hodnota DataOffset se zvýší o 1 a hodnota DataLength se sníží o 1, je-li metoda úspěšná.

Předpokládá se, že 1 znak dat zprávy je binární celé číslo bez znaménka.

Definováno v:
Třída MQMessage

Syntaxe:
integerv% = MQMessage **.ReadUnsignedByte**

Metoda ReadUTF

Tato metoda čte řetězec formátu UTF ze zprávy, počínaje od bajtu, na který se odkazuje **DataOffset**, a vrací řetězec formátu UTF jako řetězec ActiveX . Řetězec formátu UTF, který se čte, obsahuje 2 bajty dat, které označují délku řetězce, za kterými následují znaková data UTF.

Metoda selže, pokud je **MQMessage.DataLength** menší než délka řetězce, když je vydán.

DataOffset se zvýší o délku řetězce a **DataLength** se sníží o délku řetězce, je-li metoda úspěšná.

Definováno v:
Třída MQMessage

Syntaxe:

```
value$ = MQMessage.ReadUTF
```

Metoda ResizeBuffer

Tato metoda pozmění velikost úložiště, které je momentálně přiděleno interně, aby zadržoval vyrovnávací paměť dat zprávy. Poskytuje aplikaci určitou kontrolu nad automatickou správou vyrovnávací paměti tím, že v případě, že aplikace ví, že se bude zabývat velkou zprávou, může zajistit, že je alokována dostatečně velká vyrovnávací paměť. Aplikace nevyžaduje použití tohoto volání-pokud tomu tak není, bude velikost vyrovnávací paměti zvětšovat velikost vyrovnávací paměti pro automatickou správu vyrovnávací paměti.

Pokud změníte velikost vyrovnávací paměti tak, aby byla menší než aktuální hodnota MessageLength, riskujete ztrátu dat. Pokud ztratíte data, metoda vrací CompletionCode MQCC_WARNING a ReasonCode objektu MQRC_DATA_TRUNCATED.

Pokud změníte velikost vyrovnávací paměti tak, aby byla menší než hodnota vlastnosti **DataOffset** , postupujte takto:

- Vlastnost **DataOffset** se změní tak, aby ukazovala na konec nové vyrovnávací paměti.
- Vlastnost **DataLength** je nastavena na nulu.
- Vlastnost **MessageLength** se změní na novou velikost vyrovnávací paměti.

Definováno v:
Třída MQMessage

Syntaxe: *MQMessage* **.ResizeBuffer** (*Length &*)

Parametr:
Délka & Dlouhé. Velikost je povinná ve znacích.

Metoda zápisu

Zapisuje posloupnost bajtů do vyrovnávací paměti zpráv z bajtového pole na pozici, na kterou se odkazuje posunutí dat. Je-li to nutné, je délka vyrovnávací paměti (MQMessage.MQMessageLength) rozšířena tak, aby obsásla celou délku pole bajtů. Hodnota DataOffset je zvýšena o počet bajtů zapsaných, pokud byla metoda úspěšná.

Definováno v:
Třída MQMessage

Syntaxe:

```
Call MQMessage.Write(value)
```

Parametry:
data: bajtové pole nebo varianta odkazu na bajtové pole

Metoda WriteBoolean

Zapisuje 1bajtovou logickou hodnotu na aktuální pozici ve vyrovnávací paměti zpráv z 2bajtové logické hodnoty. Hodnota DataOffset se zvýší o jednu.

Definováno v:
Třída MQMessage

Syntaxe:

```
Call MQMessage.WriteBoolean(value)
```

Parametr:
hodnota: Boolean (2-bajty). Hodnota, která má být zapsána.

Metoda WriteByte

Tato metoda vezme podepsanou 2bajtovou celočíselnou hodnotu a zapíše ji do vyrovnávací paměti dat zprávy jako jednobajtové binární číslo na pozici, na kterou se odkazuje DataOffset. Pokud je to nutné, nahradí data již na pozici ve vyrovnávací paměti a rozšíří délku vyrovnávací paměti (MQMessage.MessageLength).

Hodnota DataOffset se zvýší o jednu, pokud je metoda úspěšná.

Uvedená hodnota by měla být v rozsahu -128 až 127. Pokud tomu tak není, vrátí se metoda CompletionCode MQCC_FAILED a ReasonCode MQRC_WRITE_VALUE_ERROR.

Definováno v: třídě MQMessage

Syntaxe:

```
Call MQMessage.WriteByte(value%)
```

Parametr: *value%* Integer. Hodnota, která má být zapsána.

Metoda WriteDecimal2

Zapisuje se 2 bajtové celé číslo jako 2bajtové dekadické číslo se znaménkem. Hodnota DataOffset je zvýšena o dvě.

Definováno v:
Třída MQMessage

Syntaxe:

```
Call MQMessage.WriteDecimal2(value%)
```

Parametr:
hodnota% celé číslo. Hodnota, která má být zapsána.

Metoda WriteDecimal4

Zapisuje se 4bajtové celé číslo se znaménkem jako 4bajtové dekadické číslo. Hodnota DataOffset je zvýšena o čtyři.

Definováno v:
Třída MQMessage

Syntaxe:

```
Call MQMessage.WritedDecimal4(value&)
```

Parametr:
hodnota & Long. Hodnota, která má být zapsána.

Metoda WriteDouble

Tato metoda vezme hodnotu 8bajtového čísla s pohyblivou řádovou čárkou a zapíše ji do vyrovnávací paměti dat zprávy jako 8bajtové číslo s pohyblivou řádovou čárkou, které začíná na pozici, na kterou se odkazuje hodnota DataOffset. Nahrazuje veškerá data již na těchto pozicích ve vyrovnávací paměti a v případě potřeby rozšiřuje délku vyrovnávací paměti (MQMessage.MessageLength).

Hodnota DataOffset je zvýšena o 8, je-li metoda úspěšná.

Metoda se převede na reprezentaci plovoucí řádové čárky zadané vlastností MQMessage.Encoding . *Konverze do formátu System/360 není podporována.*

Definováno v: třídě MQMessage

Syntaxe:

```
Call MQMessage.WriteDouble(value#)
```

Parametr:
Hodnota typu Double. Hodnota, která má být zapsána.

Metoda WriteDouble4

Popis nastavení příkazu ReadDouble4 a WriteDouble4 v místě ReadFloat a WriteFloatviz “Metoda ReadDouble4” na stránce 613 .

Tato metoda vezme hodnotu 8bajtového čísla s pohyblivou řádovou čárkou a zapíše ji do vyrovnávací paměti dat zprávy jako 4bajtové číslo s pohyblivou řádovou čárkou počínaje pozicí, na kterou se odkazuje hodnota DataOffset.

Hodnota DataOffset je zvýšena o 4, je-li metoda úspěšná.

Nahrazuje veškerá data již na těchto pozicích ve vyrovnávací paměti a v případě potřeby rozšiřuje délku vyrovnávací paměti (MQMessage.MessageLength).

Metoda se převede na reprezentaci plovoucí řádové čárky zadané vlastností MQMessage.Encoding . *Konverze do formátu System/360 není podporována.*

Definováno v: třídě MQMessage

Syntaxe:

```
Call MQMessage.WriteDouble4(value#)
```

Parametr: *value#* Double. Hodnota, která má být zapsána.

Metoda WriteFloat

Tato metoda vezme hodnotu 4 bajtového čísla s pohyblivou řádovou čárkou a zapíše ji do vyrovnávací paměti dat zprávy jako 4bajtové číslo s pohyblivou řádovou čárkou, které začíná na znaku, na který se odkazuje hodnota DataOffset. Nahrazuje veškerá data již na těchto pozicích ve vyrovnávací paměti a v případě potřeby rozšiřuje délku vyrovnávací paměti (MQMessage.MessageLength).

Hodnota DataOffset je zvýšena o 4, je-li metoda úspěšná.

Metoda se převede do binární reprezentace zadané vlastností MQMessage.Encoding . *Konverze do formátu System/360 není podporována.*

Definováno v: třídě MQMessage

Syntaxe:

```
Call MQMessage.WriteFloat(value!)
```

Parametr *hodnota!* Float. Hodnota, která má být zapsána.

Metoda WriteInt2

Tato metoda je identická s metodou WriteShort .

Syntaxe:

```
Call MQMessage.WriteInt2(value%)
```

Parametr *hodnota%* celé číslo. Hodnota, která má být zapsána.

Metoda WriteInt4

Tato metoda je identická s metodou WriteLong .

Syntaxe:

```
Call MQMessage.WriteInt4(value&)
```

Parametr *hodnota &* Long. Hodnota, která má být zapsána.

Metoda WriteLong

Tato metoda vezme podepsanou 4bajtovou celočíselnou hodnotu a zapíše ji do vyrovnávací paměti dat zprávy jako 4bajtové binární číslo začínající na bajtu, na který se odkazuje hodnota DataOffset. Nahrazuje veškerá data již na těchto pozicích ve vyrovnávací paměti a v případě potřeby rozšiřuje délku vyrovnávací paměti (MQMessage.MessageLength).

Hodnota DataOffset je zvýšena o 4, je-li metoda úspěšná.

Metoda se převede do binární reprezentace zadané vlastností MQMessage.Encoding .

Definováno v: třídě MQMessage

Syntaxe:

```
Call MQMessage.WriteLong(value&)
```

Parametr *hodnota &* Long. Hodnota, která má být zapsána.

Metoda WriteNullTerminatedString

Tato metoda provádí normální WriteString a vycpává zbývající bajty do zadané délky s hodnotou null. Pokud se počet bajtů zapsaných počátečním řetězcem zápisu rovná zadané délce, nezapíší se žádné hodnoty null. Pokud počet bajtů překročí uvedenou délku, bude nastavena chyba (kód příčiny MQRC_WRITE_VALUE_ERROR).

Hodnota DataOffset se zvýší o určenou délku, pokud je metoda úspěšná.

Definováno v: třídě MQMessage

Syntaxe:

```
Call MQMessage.WriteNullTerminatedString(value$, length&)
```

Parametry:
hodnotu $String. Hodnota, která má být zapsána.

délka & Long. Délka pole řetězce v bajtech.

Metoda WriteShort

Tato metoda vezme podepsanou 2bajtovou celočíselnou hodnotu a zapíše ji do vyrovnávací paměti dat zprávy jako 2bajtové binární číslo začínající na bajtu, na který se odkazuje DataOffset. Nahrazuje veškerá data již na těchto pozicích ve vyrovnávací paměti a v případě potřeby prodlouží délku vyrovnávací paměti (MQMessage.MessageLength).

Hodnota DataOffset se zvýší o 2, pokud je metoda úspěšná.

Metoda se převede do binární reprezentace zadané vlastností MQMessage.Encoding .

Definováno v: třídě MQMessage

Syntaxe:

```
Call MQMessage.WriteShort(value%)
```

Parametr *hodnota%* celé číslo. Hodnota, která má být zapsána.

Metoda WriteString

Tato metoda vezme řetězec ActiveX a zapíše jej do vyrovnávací paměti dat zprávy začínající na bajtu, na který se odkazuje hodnota DataOffset. Nahrazuje veškerá data již na těchto pozicích ve vyrovnávací paměti a v případě potřeby prodlouží délku vyrovnávací paměti (MQMessage.MessageLength).

Hodnota DataOffset je zvětšena o délku řetězce v bajtech, pokud metoda uspěje.

Metoda převede znaky na kódovou stránku určenou vlastností MQMessage.CharacterSet .

Definováno v: třídě MQMessage

Syntaxe:

```
Call MQMessage.WriteString(value$)
```

Parametr *hodnota $* Řetězec. Hodnota, která má být zapsána.

Metoda WriteUInt2

Tato metoda vezme podepsanou 4bajtovou celočíselnou hodnotu a zapíše ji do vyrovnávací paměti dat zprávy jako 2bajtové binární číslo bez znaménka začínající na bajtu, na který odkazuje hodnota DataOffset. Nahrazuje veškerá data již na těchto pozicích ve vyrovnávací paměti a v případě potřeby rozšiřuje délku vyrovnávací paměti (MQMessage.MessageLength).

Hodnota DataOffset se zvýší o 2, pokud je metoda úspěšná.

Metoda se převede do binární reprezentace zadané vlastností MQMessage.Encoding . Uvedená hodnota by měla být v rozsahu 0 až 2 * *16-1. Pokud se nejedná o metodu s návratem CompletionCode MQCC_FAILED a ReasonCode MQRC_WRITE_VALUE_ERROR.

Definováno v: třídě MQMessage

Syntaxe:

```
Call MQMessage.WriteUInt2(value&)
```

Parametr *hodnota &* Long. Hodnota, která má být zapsána.

WriteUnsignedBytová metoda

Tato metoda vezme podepsanou 2bajtovou celočíselnou hodnotu a zapíše ji do vyrovnávací paměti dat zprávy jako 1-bajtové binární číslo bez znaménka začínající na znaku, na který se odkazuje hodnota DataOffset. Nahrazuje veškerá data již na těchto pozicích ve vyrovnávací paměti a v případě potřeby rozšiřuje délku vyrovnávací paměti (MQMessage.MessageLength).

Hodnota DataOffset je zvýšena o 1, je-li metoda úspěšná.

Uvedená hodnota by měla být v rozsahu 0 až 255. Pokud se nejedná o metodu s návratem CompletionCode MQCC_FAILED a ReasonCode MQRC_WRITE_VALUE_ERROR.

Definováno v:
Třída MQMessage

Syntaxe:

```
Call MQMessage.WriteUnsignedByte(value%)
```

Parametr *hodnota%* celé číslo. Hodnota, která má být zapsána.

Metoda WriteUTF

Tato metoda vezme řetězec ActiveX a zapíše jej do vyrovnávací paměti dat zprávy na aktuální pozici ve formátu UTF. Zapsaných dat se skládá z 2bajtové délky následované znakovými daty. Hodnota DataOffset je zvětšena o délku řetězce, pokud metoda uspěje.

Definováno v:
Třída MQMessage

Syntaxe: Volání *MQMessage*. **WriteUTF** (hodnota$)

Parametr:
hodnota $String. Hodnota, která má být zapsána.

Třída voleb MQPutMessage

Tato třída zapouzdřuje různé volby, které řídí činnost vkládání zprávy do fronty produktu IBM MQ .

Obsažení

Třída voleb MQPutMessageje obsažena ve třídě MQSession.

VYTVOŘENÍ

Volba **Nový** vytvoří nový objekt voleb MQPutMessagea nastaví všechny jeho vlastnosti na počáteční hodnoty.

Případně můžete použít metodu AccessPutMessageOptions třídy MQSession.

Syntaxe

Prom *pmo* **Jako Nový MQPutMessage** nebo

Nastavit *pmo* = **Nový VolbyMQPutMessage**

Vlastnosti

- “Vlastnost CompletionCode” na stránce 621.
- “Vlastnost Options” na stránce 622.
- “Vlastnost ReasonCode” na stránce 622.
- “Vlastnost ReasonName” na stránce 622.
- “Vlastnost RecordFields” na stránce 622.
- “Vlastnost ResolvedQueueManagerName” na stránce 622.
- “Vlastnost názvu ResolvedQueue” na stránce 623.

Metody

- “Metoda ClearErrorCodes” na stránce 623.

Vlastnost CompletionCode

Pouze pro čtení. Vrací kód dokončení nastavený podle poslední metody nebo přístupu k vlastnosti, který byl vydán proti objektu.

Definováno v: třídě voleb MQPutMessage

Datový typ: Long

Hodnoty:

- MQCC_OK
- VAROVÁNÍ MQCC_WARNING

- SELHÁNÍ MQCC_FAILED

Syntaxe: Chcete-li získat: *completioncode & = PutOpts* **.CompletionCode**

Vlastnost Options

Čtení a zápis. Pole Volby MQPMO. Počáteční hodnota tohoto pole je MQPMO_NONE. Další informace naleznete v tématu Volby MQPMO.

Definováno v: třídě voleb MQPutMessage.

Datový typ: Long

Syntaxe: Chcete-li získat: *volby & = PutOpts* **.Volby**

Chcete-li nastavit: *PutOpts* **.Volby** = *volby &*

Volby MQPMO_PASS_IDENTITY_CONTEXT a MQPMO_PASS_ALL_CONTEXT nejsou podporovány.

Vlastnost ReasonCode

Pouze pro čtení. Vrátí kód příčiny nastavený posledním voláním metody nebo vlastností, které byly vydány pro daný objekt.

Definováno v: třídě voleb MQPutMessage

Datový typ: Long

Hodnoty:

- Viz termín completion API a kódy příčin.

Syntaxe: Chcete-li získat následující informace: *reasoncode & = PutOpts* **.ReasonCode**

Vlastnost ReasonName

Pouze pro čtení. Vrátí symbolický název posledního kódu příčiny. Příklad: "MQRC_QMGR_NOT_AVAILABLE".

Definováno v: třídě voleb MQPutMessage

Datový typ: Řetězec

Hodnoty:

- Viz termín completion API a kódy příčin.

Syntaxe: Chcete-li získat: *reasonname $= PutOpts* **.ReasonName**

Vlastnost RecordFields

Čtení a zápis. Příznaky označující, která pole mají být při vložení zprávy do distribučního seznamu přizpůsobena pro jednotlivé fronty. Počáteční hodnota je nula.

Tato vlastnost odpovídá příznaky PutMsgRecFields ve struktuře MQPMO rozhraní MQI MQI. V rozhraní MQI tyto příznaky řídí, která pole (ve struktuře MQPMR) jsou přítomna a používána operací MQPUT. V objektu Volby MQPutMessagejsou tato pole vždy přítomná a parametry proto ovlivňují pouze ta pole, která jsou použita vložením.

Definováno v:
Třída voleb MQPutMessage

Datový typ:
Dlouhý

Syntaxe: Chcete-li získat: *recordfields & = PutOpts* **.RecordFields**

Chcete-li nastavit: *PutOpts*. **RecordFields** = *recordfields &*

Vlastnost ResolvedQueueManagerName

Pouze pro čtení. Pole názvu MQPMO ResolvedQMgr. Podrobnosti najdete v tématu ResolvedQMgrName (MQCHAR48) . Počáteční hodnota je prázdná.

Definováno v: třídě voleb MQPutMessage

Typ dat: Řetězec 48 znaků

Syntaxe: Chcete-li získat: *qmgr $= PutOpts* **.ResolvedQueueManagerName**

Vlastnost názvu ResolvedQueue

Pouze pro čtení. Pole MQPMO ResolvedQName . Podrobnosti viz ResolvedQName (MQCHAR48) . Počáteční hodnota je prázdná.

Definováno v: třídě voleb MQPutMessage

Typ dat: Řetězec 48 znaků

Syntaxe: Chcete-li získat: *qname $= PutOpts* **.ResolvedQueueName**

Metoda ClearErrorCodes

Resetuje parametr CompletionCode na MQCC_OK a ReasonCode na hodnotu MQRC_NONE jak pro třídu voleb MQPutMessage, tak pro třídu MQSession.

Definováno v:
Třída voleb MQPutMessage

Syntaxe:
Volání *PutOpts* **.ClearErrorCodes ()**

Třída voleb MQGetMessage

Tato třída zapouzdřuje různé volby, které řídí akci získání zprávy z fronty produktu IBM MQ .

Obsažení

Třída voleb MQGetMessageje obsažena ve třídě MQSession.

VYTVOŘENÍ

Volba **Nový** vytvoří nový objekt voleb MQGetMessagea nastaví všechny její vlastnosti na počáteční hodnoty.

Případně můžete použít metodu AccessGetMessageOptions třídy MQSession.

Vlastnosti

Metody

Syntaxe

Dim *gmo* **Jako nové volby MQGetMessage** nebo

Nastavit *gmo* = **Nové volby MQGetMessage**

Vlastnost CompletionCode

Pouze pro čtení. Vrací kód dokončení nastavený podle poslední metody nebo přístupu k vlastnosti, který byl vydán proti objektu.

Definováno v: třídě voleb MQGetMessage.

Datový typ: Long

Hodnoty:

- MQCC_OK
- VAROVÁNÍ MQCC_WARNING
- SELHÁNÍ MQCC_FAILED

Syntaxe: Chcete-li získat: *completioncode & = GetOpts* **.CompletionCode**

Vlastnost MatchOptions

Čtení a zápis. Volby, které řídí kritéria výběru použitá pro MQGET. Počáteční hodnota je MQMO_MATCH_MSG_ID + MQMO_MATCH_CORREL_ID.

Definováno v:
Třída voleb MQGetMessage

Datový typ:
Dlouhý

Hodnoty:
Viz MatchOptions (MQLONG).

Syntaxe: Chcete-li získat: *matchoptions & = GetOpts*. **MatchOptions**

Chcete-li nastavit: *GetOpts*. **MatchOptions** = *matchoptions &*

Vlastnost Options

Čtení a zápis. Pole Volby MQGMO. Podrobnosti viz Volby . Počáteční hodnota je MQGMO_NO_WAIT.

Definováno v: třídě voleb MQGetMessage.

Datový typ: Long

Syntaxe: Chcete-li získat: *volby & = GetOpts* **.Volby** Chcete-li nastavit: *GetOpts* **.Volby** = *volby &*

Vlastnost ReasonCode

Pouze pro čtení. Vrátí kód příčiny nastavený posledním voláním metody nebo vlastností, které byly vydány pro daný objekt.

Definováno v: třídě voleb MQGetMessage

Datový typ: Long

Hodnoty:

- Viz termín completion API a kódy příčin.

Syntaxe: Chcete-li získat následující informace: *reasoncode & = GetOpts* **.ReasonCode**

Vlastnost ReasonName

Pouze pro čtení. Vrátí symbolický název posledního kódu příčiny. Příklad: "MQRC_QMGR_NOT_AVAILABLE". **Definováno v:** třídě voleb MQGetMessage

Datový typ: Řetězec

Hodnoty:

- Viz termín completion API a kódy příčin.

Syntaxe: Chcete-li získat: *reasonname $= MQGetMessageOptions* **.ReasonName**

Vlastnost názvu ResolvedQueue

Pouze pro čtení. Pole MQGMO ResolvedQName . Podrobnosti viz ResolvedQName (MQCHAR48) . Počáteční hodnota je prázdná.

Definováno v: třídě voleb MQGetMessage

Typ dat: Řetězec 48 znaků

Syntaxe: Chcete-li získat: *qname $= GetOpts* **.ResolvedQueueName**

Vlastnost WaitInterval

Čtení/zápis. Pole MQGMO WaitInterval . Maximální doba v milisekundách, po kterou komponenta Get čeká na vhodnou zprávu-pokud byla akce čekání požadována vlastností Options. Toto pole má počáteční hodnotu 0. Podrobné informace o volbách MQGMO naleznete v tématu MQGMO.

Definováno v: třídě voleb MQGetMessage

Datový typ: Long

Syntaxe: Chcete-li získat: *wait & = GetOpts* **.WaitInterval**

Pro nastavení: *GetOpts* **.WaitInterval** = wait &

Metoda ClearErrorCodes

Resetuje parametr CompletionCode na MQCC_OK a ReasonCode na MQRC_NONE jak pro třídu voleb MQGetMessage, tak pro třídu MQSession.

Definováno v:
Třída voleb MQGetMessage

Syntaxe:
Volání *GetOpts* **.ClearErrorCodes ()**

Třída MQDistributionList

Tato třída zapouzdřuje kolekci front-lokálních, vzdálených nebo aliasů pro výstup.

VYTVOŘENÍ

new vytvoří nový objekt MQDistributionList .

Případně můžete použít metodu AddDistributionList třídy MQQueueManager .

Vlastnosti

- "Vlastnost ID AlternateUser" na stránce 626
- "Vlastnost CloseOptions" na stránce 626
- "Vlastnost CompletionCode" na stránce 626
- "Vlastnost ConnectionReference" na stránce 627
- "Vlastnost FirstDistributionListItem" na stránce 627

Metoda

Syntaxe

Dim *seznam_distR.* **A s novými MQDistributionList** nebo **Set** *distlist* = **New MQDistributionList**

Vlastnost ID AlternateUser

Čtení a zápis. Alternativní ID uživatele použité pro ověření přístupu k seznamu front, když jsou otevřeny.

Definováno v:
Třída MQDistributionList

Datový typ:
Řetězec 12 znaků

Syntaxe: Chcete-li získat: *altuser $= MQDistributionList*. **AlternateUserid**

Chcete-li nastavit: *MQDistributionList*. **AlternateUserId** = *altuser $*

Vlastnost CloseOptions

Čtení a zápis. Volby používané k řízení toho, co se stane, když se distribuční seznam uzavře. Počáteční hodnota je MQCO_NONE.

Definováno v:
Třída MQDistributionList

Datový typ:
Dlouhý

Hodnoty:

- MQCO_NONE
- MQCO_DELETE
- MQCO_DELETE_PURGE

Syntaxe: Chcete-li získat následující informace: *closeopt & = MQDistributionList*. **CloseOptions**

Chcete-li nastavit: *MQDistributionList*. **CloseOptions** = *closeopt &*

Vlastnost CompletionCode

Pouze pro čtení. Kód dokončení nastavený pomocí poslední metody nebo přístupu k vlastnosti vydaného pro objekt.

Definováno v:
Třída MQDistributionList

Datový typ:
Dlouhý

Hodnoty:

- MQCC_OK
- VAROVÁNÍ MQCC_WARNING
- SELHÁNÍ MQCC_FAILED

Syntaxe: Chcete-li získat: *completioncode & = MQDistributionList*. **CompletionCode**

Vlastnost ConnectionReference

Čtení a zápis. Správce front, do kterého patří distribuční seznam.

Definováno v:
Třída MQDistributionList

Datový typ:
MQQueueManager

Syntaxe: Chcete-li získat: *set queuemanager = MQDistributionList*. **ConnectionReference**

Nastavení: *set MQDistributionList*. **ConnectionReference** = *queuemanager*

Vlastnost FirstDistributionListItem

Pouze pro čtení. První objekt položky rozdělovníku přidružený k rozdělovníku.

Definováno v:
Třída MQDistributionList

Datový typ:
Položka MQDistributionList

Hodnoty:

Syntaxe: Chcete-li získat: *set distributionlistitem = MQDistributionList*. **FirstDistributionListItem**

Vlastnost IsOpen

Pouze pro čtení.

Definováno v:
Třída MQDistributionList

Datový typ:
Logická hodnota

Hodnoty:

- TRUE (-1)
- FALSE (0)

Syntaxe: Chcete-li získat: *IsOpen = MQDistributionList*. **IsOpen**

Vlastnost OpenOptions

Čtení a zápis. Volby, které se mají použít při otevření distribučního seznamu.

Definováno v:
Třída MQDistributionList

Datový typ:
Dlouhý

Hodnoty:
Viz Volby MQPMO.

Syntaxe: Chcete-li získat následující informace: *openopt & = MQDistributionList*. **OpenOptions**

Chcete-li nastavit: *MQDistributionList.* **OpenOptions** = openopt &

Vlastnost ReasonCode

Pouze pro čtení. Kód příčiny nastavený pomocí poslední metody nebo přístupu k vlastnosti vydaného pro objekt.

Definováno v:
Třída MQDistributionList

Datový typ:
Dlouhý

Hodnoty:
Viz termín completion API a kódy příčin.

Syntaxe: Chcete-li získat následující informace: *reasoncode & = MQDistributionList*. **ReasonCode**

Vlastnost ReasonName

Pouze pro čtení. Symbolický název pro ReasonCode. Například "MQRC_QMGR_NOT_AVAILABLE".

Definováno v:
Třída MQDistributionList

Datový typ:
Řetězec

Hodnoty:
Viz termín completion API a kódy příčin.

Syntaxe: Chcete-li získat: *reasonname $= MQDistributionList*. **ReasonName**

Metoda AddDistributionListItem

Tato metoda se používá k vytvoření nového objektu `MQDistributionListItem` a k jeho přidružení k objektu seznamu distribuce. Parametr názvu fronty je povinný.

Tato metoda vloží novou položku distribučního seznamu jako první položku v existujícím seznamu. Specificky tato metoda vytváří následující konfiguraci:

- V rozdělovníku nastaví vlastnost **`FirstDistributionListItem`** tak, aby ukazovala na novou položku rozdělovníku.
- V nové položce distribučního seznamu nastavuje následující vlastnosti:
  - Nastavuje vlastnost **`DistributionList`** tak, aby ukazovala na distribuční seznam.
  - Nastavuje vlastnost **`PreviousDistributionListItem`** na hodnotu null.
  - Nastavuje vlastnost **`NextDistributionListItem`** tak, aby ukazovala na položku distribučního seznamu, která byla dříve jako první, nebo na hodnotu null, pokud v seznamu neexistovaly žádné předchozí položky.

Tuto metodu nelze použít k přidání nové položky, je-li distribuční seznam otevřený.

Definováno v:
Třída MQDistributionList

Syntaxe: set distributionlistitem = *MQDistributionList* **.AddDistributionListItem** (QName$, QMgrName$)

Parametry:

Řetězec *QName$* . Název fronty produktu IBM MQ .

Řetězec *QMgrName$* String. Název správce front produktu IBM MQ .

Metoda ClearErrorCodes

Resetuje parametr CompletionCode na MQCC_OK a ReasonCode na MQRC_NONE jak pro třídu MQDistributionList , tak pro třídu MQSession.

Definováno v:
Třída MQDistributionList

Syntaxe:

```
Call MQDistributionList.ClearErrorCodes()
```

Metoda Close

Zavře distribuční seznam s použitím aktuální hodnoty voleb Zavřít.

Definováno v:
Třída MQDistributionList

Syntaxe:
`Call MQDistributionList. Close ()`

Otevřít metodu

Otevře všechny fronty určené vlastnostmi **QueueName** a (kde je to vhodné) **QueueManagerName** položek rozdělovníku přidružených k aktuálnímu objektu pomocí aktuální hodnoty ID AlternateUserId.

Definováno v:
Třída MQDistributionList

Syntaxe:

```
Call MQDistributionList.Open()
```

metoda PUT

Umístí zprávu na každou z front uvedených v položkách rozdělovníku přidružených k rozdělovníku.

Definováno v:

Třída MQDistributionList

Syntaxe

Volejte funkci MQDistributionList. **Vložit** (Zpráva, volby PutMsg&)

Parametry

Objekt *Message* MQMessage představující zprávu, která má být vložena.

Objekt *PutMsgOptions* MQPutMessageOptions obsahující volby pro řízení operace put. Není-li tento parametr zadán, použijí se výchozí volby PutMessage.

Tato metoda přijímá objekt MQMessage jako parametr. Jako výsledek této metody lze změnit následující vlastnosti položky seznamu distribucí:

- CompletionCode
- ReasonCode
- ReasonName
- MessageId
- MessageIdHex
- CorrelationId
- CorrelationIdHexadecimální číslo
- GroupId
- GroupIdHexadecimální
- Zpětná vazba

- AccountingToken
- AccountingTokenHex

Třída položek MQDistributionList

Tato třída zapouzdřuje struktury MQOR, MQRR a MQPMR a sdružuje je s vlastnícím distribučním seznamem.

VYTVOŘENÍ

Použití metody AddDistributionListItem třídy MQDistributionList

Vlastnosti

Metody

- “Vlastnost AccountingToken” na stránce 631.
- “AccountingTokenHexadecimální vlastnost” na stránce 631.
- “Vlastnost CompletionCode” na stránce 631.
- “Vlastnost CorrelationId” na stránce 632.
- “CorrelationIdHexadecimální vlastnost” na stránce 632.
- “Vlastnost DistributionList” na stránce 632.
- “Vlastnost Feedback” na stránce 632.
- “Vlastnost GroupId” na stránce 632.
- “GroupIdHexadecimální vlastnost” na stránce 633.
- “Vlastnost MessageId” na stránce 633.
- “MessageIdHexadecimální vlastnost” na stránce 633.
- “Vlastnost NextDistributionListItem” na stránce 633.
- “Vlastnost PreviousDistributionListItem” na stránce 634.
- “Vlastnost názvu QueueManager” na stránce 634.
- “Vlastnost QueueName” na stránce 634.
- “Vlastnost ReasonCode” na stránce 634.
- “Vlastnost ReasonName” na stránce 634.
- “Metoda ClearErrorCodes” na stránce 635.

Vlastnosti:

- Vlastnost AccountingToken
- AccountingTokenHexadecimální vlastnost
- Vlastnost CompletionCode
- Vlastnost CorrelationId
- CorrelationIdHexadecimální vlastnost
- Vlastnost DistributionList
- Vlastnost Feedback
- Vlastnost GroupId
- GroupIdHexadecimální vlastnost
- Vlastnost MessageId
- MessageIdHexadecimální vlastnost

- Vlastnost NextDistributionListItem
- Vlastnost PreviousDistributionListItem
- Vlastnost názvu QueueManager
- Vlastnost QueueName
- Vlastnost ReasonCode
- Vlastnost ReasonName

Metody:

- Metoda ClearErrorCodes

Vytvoření:

Použití metody AddDistributionListItem třídy MQDistributionList

Vlastnost AccountingToken

Čtení a zápis. Hodnota AccountingToken bude zahrnuta do zprávy MQPMR zprávy při vložení do fronty. Jeho počáteční hodnota je všechny hodnoty null.

Definováno v:
Třída položek MQDistributionList

Datový typ:
Řetězec 32 znaků

Syntaxe: Chcete-li získat: *accountingtoken $= MQDistributionListItem*. **AccountingToken**

Chcete-li nastavit: *MQDistributionListItem*. **AccountingToken** = *accountingtoken $*

AccountingTokenHexadecimální vlastnost

Čtení a zápis. Hodnota AccountingToken bude zahrnuta do zprávy MQPMR zprávy při vložení do fronty.

Každé dva znaky řetězce reprezentují hexadecimální ekvivalent jednoho znaku ASCII. Například dvojice znaků "6" a "1" představují jediný znak "A", dvojice znaků "6" a "2" představují jediný znak "B" a tak dále.

Je třeba zadat 64 platných hexadecimálních znaků.

Jeho počáteční hodnota je "0 ... 0".

Definováno v:
Třída položek MQDistributionList

Datový typ:
Řetězec obsahující 64 hexadecimálních znaků, který požaduje 32 znaků ASCII.

Syntaxe: Chcete-li získat: *accountingtokenh $= MQDistributionListItem*. **AccountingTokenHex**

Chcete-li nastavit: *MQDistributionListItem*. **AccountingTokenHex** = *accountingtokenh $* .

Vlastnost CompletionCode

Pouze pro čtení. Kód dokončení nastavený posledním otevřeným vydáním nebo požadavkem na vložení vydaného na vlastnící objekt distribučního seznamu.

Definováno v:
Třída položek MQDistributionList

Datový typ:
Dlouhý

Hodnoty:

- MQCC_OK
- VAROVÁNÍ MQCC_WARNING

- SELHÁNÍ MQCC_FAILED

Syntaxe: Chcete-li získat: *completioncode $= MQDistributionListItem*. **CompletionCode**

Vlastnost CorrelationId

Čtení a zápis. CorrelId , které má být zahrnuto do MQPMR zprávy při vložení do fronty. Jeho počáteční hodnota je všechny hodnoty null.

Definováno v:
Třída položek MQDistributionList

Datový typ:
Řetězec o délce 24 znaků

Syntaxe: Chcete-li získat: *correlid $= MQDistributionListItem*. **CorrelationId**

Chcete-li nastavit: *MQDistributionListItem*. **CorrelationId** = *correlid $*

CorrelationIdHexadecimální vlastnost

Čtení a zápis. CorrelId , které má být zahrnuto do MQPMR zprávy při vložení do fronty.

Každé dva znaky řetězce reprezentují hexadecimální ekvivalent jednoho znaku ASCII. Například dvojice znaků "6" a "1" představují jediný znak "A", dvojice znaků "6" a "2" představují jediný znak "B" a tak dále.

Je třeba zadat 48 platných hexadecimálních znaků.

Jeho počáteční hodnota je "0 .. 0".

Definováno v:
Třída položek MQDistributionList

Datový typ:
Řetězec 48 hexadecimálních znaků představujících 24 znaků ASCII.

Syntaxe: Chcete-li získat: *correrelidh $= MQDistributionListItem*. **CorrelationIdHexadecimální číslo**

Chcete-li nastavit: *MQDistributionListItem*. **CorrelationIdHex** = *correlidh $*

Vlastnost DistributionList

Pouze pro čtení. Distribuční seznam, se kterým je tato položka rozdělovníku přidružena.

Definováno v:
Třída položek MQDistributionList

Datový typ:
MQDistributionList

Syntaxe: Chcete-li získat: *set distributionlist = MQDistributionListItem*. **DistributionList**

Vlastnost Feedback

Čtení a zápis. Hodnota zpětné vazby, která má být zahrnuta do zprávy MQPMR zprávy při vložení do fronty.

Definováno v:
Třída položek MQDistributionList

Datový typ:
Dlouhý

Hodnoty:
Viz Zpětná vazba (MQLONG).

Syntaxe: Chcete-li získat následující informace: *feedback & = MQDistributionListItem*. **Zpětná vazba**

Chcete-li nastavit: *MQDistributionListItem*. **Zpětná vazba** = *feedback &*

Vlastnost GroupId

Čtení a zápis. Hodnota GroupId , která má být zahrnuta do zprávy MQPMR zprávy při vložení do fronty. Jeho počáteční hodnota je všechny hodnoty null.

Definováno v:
Třída položek MQDistributionList

Datový typ:
Řetězec o délce 24 znaků

Syntaxe: Chcete-li získat: *groupid $= MQDistributionListItem*. **GroupId**

Chcete-li nastavit: *MQDistributionListItem*. **GroupId** = *groupid $*

GroupIdHexadecimální vlastnost

Čtení a zápis. Hodnota GroupId , která má být zahrnuta do zprávy MQPMR zprávy při vložení do fronty.

Každé dva znaky řetězce reprezentují hexadecimální ekvivalent jednoho znaku ASCII. Například dvojice znaků "6" a "1" představují jediný znak "A", dvojice znaků "6" a "2" představují jediný znak "B" a tak dále.

Je třeba zadat 48 platných hexadecimálních znaků.

Jeho počáteční hodnota je "0 .. 0".

Definováno v:
Třída položek MQDistributionList

Datový typ:
Řetězec 48 hexadecimálních znaků reqpresenting 24 znaků ASCII.

Syntaxe: Chcete-li získat: *groupidh $= MQDistributionListItem*. **GroupIdHex**

Chcete-li nastavit: *MQDistributionListItem*. **GroupIdHex** = *groupidh $*

Vlastnost MessageId

Čtení a zápis. Hodnota MessageId , která má být zahrnuta do zprávy MQPMR zprávy při vložení do fronty. Jeho počáteční hodnota je všechny hodnoty null.

Definováno v:
Třída položek MQDistributionList

Datový typ:
Řetězec o délce 24 znaků

Syntaxe: Chcete-li získat: *messageid $= MQDistributionListItem*. **MessageId**

Chcete-li nastavit: *MQDistributionListItem*. **MessageId** = *messageid $* .

MessageIdHexadecimální vlastnost

Čtení a zápis. Hodnota MessageId , která má být zahrnuta do zprávy MQPMR zprávy při vložení do fronty.

Každé dva znaky řetězce reprezentují hexadecimální ekvivalent jednoho znaku ASCII. Například dvojice znaků "6" a "1" představují jediný znak "A", dvojice znaků "6" a "2" představují jediný znak "B" a tak dále.

Je třeba zadat 48 platných hexadecimálních znaků.

Jeho počáteční hodnota je "0 .. 0".

Definováno v:
Třída položek MQDistributionList

Datový typ:
Řetězec 48 hexadecimálních znaků představujících 24 znaků ASCII.

Syntaxe: Chcete-li získat: *messagesh $= MQDistributionListItem*. **MessageIdHex**

Chcete-li nastavit: *MQDistributionListItem*. **MessageIdHex** = *messageidh $*

Vlastnost NextDistributionListItem

Pouze pro čtení. Další objekt položky rozdělovníku přidružený ke stejnému rozdělovníku.

Definováno v:
Třída položek MQDistributionList

Datový typ:
Položka MQDistributionList

Syntaxe: Chcete-li získat: *set distributionlistitem = MQDistributionListItem.* **NextDistributionListItem**

Vlastnost PreviousDistributionListItem

Pouze pro čtení. Předchozí objekt položky seznamu distribuce přidružený ke stejnému rozdělovníku.

Definováno v:
Třída položek MQDistributionList

Datový typ:
Položka MQDistributionList

Syntaxe: Chcete-li získat: *set distributionlistitem = MQDistributionListItem*. **PreviousDistributionListItem**

Vlastnost názvu QueueManager

Čtení a zápis. Název správce front produktu IBM MQ .

Definováno v:
Třída položek MQDistributionList

Datový typ:
Řetězec 48 znaků.

Syntaxe: Chcete-li získat: *qmname $= MQDistributionListItem*. **QueueManagerName**

Chcete-li nastavit: *MQDistributionListItem*. **QueueManagerNázev** = *qmname $*

Vlastnost QueueName

Čtení a zápis. Název fronty produktu IBM MQ .

Definováno v:
Třída položek MQDistributionList

Datový typ:
Řetězec 48 znaků.

Syntaxe: Chcete-li získat: *qname $= MQDistributionListItem*. **QueueName**

Chcete-li nastavit: *MQDistributionListItem*. **QueueName** = *qname $*

Vlastnost ReasonCode

Pouze pro čtení. Kód příčiny nastavený posledním otevření nebo vložení vydaného na vlastnící objekt distribučního seznamu.

Definováno v:
Třída položek MQDistributionList

Datový typ:
Dlouhý

Hodnoty:
Viz termín completion API a kódy příčin.

Syntaxe: Chcete-li získat:

```
reasoncode& = MQDistributionListItem.ReasonCode
```

Vlastnost ReasonName

Pouze pro čtení. Symbolický název pro ReasonCode. Například "MQRC_QMGR_NOT_AVAILABLE".

Definováno v:
Třída položek MQDistributionList

Datový typ:
Řetězec

Hodnoty:
Viz termín completion API a kódy příčin.

Syntaxe: Chcete-li získat: *reasonname $= MQDistributionListItem*. **ReasonName**

Metoda ClearErrorCodes

Resetuje parametr CompletionCode na MQCC_OK a ReasonCode na MQRC_NONE jak pro třídu položek MQDistributionList, tak pro třídu MQSession.

Definováno v:
Třída položek MQDistributionList

Syntaxe:

```
Call MQDistributionListItem.ClearErrorCodes
```

Trasování IBM MQ tříd automatizace pro ActiveX

Informace o trasovacím prostředku poskytované pro IBM MQ Automation Classes for ActiveX, běžné nástrahy a pomoc s tím, jak se jim vyhnout.

V 9.0.0 V produktu IBM MQ 9.0je podpora produktu MicrosoftActive X zamítnuta. Třídy IBM MQ pro .NET jsou doporučenými náhradními technologiemi. Další informace viz Vývoj aplikací .NET.

Tento oddíl vysvětluje poskytovanou službu trasování a podrobnosti o úskalí, které jim pomohou vyhnout se jim:

- “Řízení trasování pro IBM MQ Automation Classes for ActiveX” na stránce 635
- “Pokud se nedaří skript IBM MQ Automation Classes for ActiveX” na stránce 636
- “Kód příčiny IBM MQ Automation Classes for ActiveX .” na stránce 637
- “Nástroj na úrovni kódu” na stránce 641

Řízení trasování pro IBM MQ Automation Classes for ActiveX

Produkt IBM MQ Automation Classes for ActiveX (MQAX) obsahuje trasovací prostředek, který pomáhá organizaci služeb identifikovat to, co se děje, když máte problém.

Zobrazuje cesty, které se podnikli při spuštění skriptu MQAX . Pokud nemáte problém, spusťte s vypnutým trasováním, abyste se vyhnuli zbytečnému používání systémových prostředků.

Existují tři proměnné prostředí, které jste nastavili pro řízení trasování:

- TRASOVÁNÍ OMQ_
- CESTOVACÍ_CESTA VYNECHÁNÍ
- OMQ_TRACE_LEVEL

Poznámka: Zadání hodnoty *any* pro proměnnou **OMQ_TRACE** přepne trasovací prostředek na hodnotu. I když nastavíte proměnnou **OMQ_TRACE** na hodnotu OFF, trasování je stále aktivní. Chcete-li vypnout trasování, neuvádějte hodnotu pro **OMQ_TRACE**.

1. Klepněte na tlačítko **Spustit**
2. Klepněte na **Ovládací panely**
3. Dvakrát klepněte na **Systém**

4. Klepněte na **Rozšířené** .
5. Klepněte na volbu **Prostředí**
6. V sekci s názvem **Uživatelské proměnné pro (jméno uživatele)**klepněte na volbu **Nový** .
7. Zadejte název proměnné a platnou hodnotu do příslušných polí a klepněte na tlačítko **OK** .
8. Klepnutím na tlačítko **OK** zavřete okno Proměnné prostředí.
9. Klepnutím na tlačítko **OK** zavřete okno Systémové vlastnosti.
10. Zavřít okno Ovládací panely

Při rozhodování o tom, kam chcete trasovací soubory zapsat, ujistěte se, že máte dostatečné oprávnění k zápisu a čtení z disku.

Je-li povoleno trasování, zpomaluje spouštění struktury MQAX, ale neovlivňuje výkon vašich prostředí ActiveX nebo IBM MQ . Pokud již trasovací soubor nepotřebujete, můžete jej odstranit.

Nelze změnit stav proměnné ODMQ_TRACE, zatímco je spuštěna funkce MQAX.

Název a adresář trasovacího souboru

Název trasovacího souboru má tvar `OMQ`*nnnnn*`.trc`, kde *nnnnn* je ID procesu ActiveX spuštěného v daném okamžiku.

Příkaz	Efekt
SET OMQ_TRACE_PATH = jednotka: \adresář	Nastavuje adresář trasování, ve kterém je zapisován trasovací soubor.
NASTAVIT CESTU OQ_TRACE_PATH =	Odstraní jakékoli existující nastavení pro adresář trasování. Není-li trasovací adresář nastaven, použije se aktuální pracovní adresář (je-li spuštěn ActiveX).
ECHO %OMQ_TRACE_PATH%	Zobrazí aktuální nastavení pro trasovací adresář v systému Windows.
SET OMQ_TRACE = xxxxxxxx	Přepne na trasování. Trasování povolíte vložením jednoho nebo více znaků za znak '='. Například: `SET OMQ_TRACE=yes` a `SET OMQ_TRACE=no`. V obou těchto příkladech je trasování povoleno. Toto nastavení je platné pouze pro jedno okno/relace.
NASTAVIT FUNKCI OQ_TRACE=	Vypne trasování.
ECHO %OMQ_TRACE%	Zobrazí obsah proměnné prostředí v systému Windows.
SET	Zobrazí obsah všech proměnných prostředí v systému Windows.
NASTAVIT PARAMETR OMQ_TRACE_LEVEL = 9	Nastavuje úroveň trasování na hodnotu 9. Hodnoty větší než 9 nevytvářejí žádné další informace v trasovacím souboru.

Pokud se nedaří skript IBM MQ Automation Classes for ActiveX

Pokud se vaše skripty automatizace IBM MQ pro skript ActiveX nezdaří, je zde několik zdrojů informací, na které se můžete podívat.

Hlášení projevů prvního selhání

Nezávisle na mechanizmu trasování, v případě neočekávaných a interních chyb, může být vytvořena zpráva o selhání prvního selhání.

Tato sestava se nachází v souboru s názvem `OMQ`*nnnnn*`.fdc`, kde *nnnnn* je číslo procesu ActiveX , který běží v daném okamžiku. Tento soubor najdete v pracovním adresáři, ze kterého jste spustili ActiveX nebo v cestě uvedené v proměnné prostředí OMQ_PATH.

Další zdroje informací

Produkt IBM MQ poskytuje různé protokoly chyb a trasovací informace v závislosti na použité platformě. Podívejte se do protokolu událostí aplikace Windows .

Kód příčiny IBM MQ Automation Classes for ActiveX .

Kódy příčiny pro IBM MQ Automation Classes for ActiveX (MQAX), které se mohou vyskytnout spolu s kódem příčiny IBM MQ MQI.

Kromě těch, které jsou zdokumentovány pro rozhraní IBM MQ MQI, mohou být kromě uvedených kódů příčiny uvedeny také následující kódy příčiny. Další kódy najdete v protokolu událostí aplikace IBM MQ .

Kód příčiny	Vysvětlení
MQRC_LIBRARY_LOAD_ERROR (6000)	Jednu nebo více knihoven IBM MQ nebylo možné načíst. Zkontrolujte, že všechny knihovny IBM MQ jsou ve správné vyhledávací cestě v systému, který používáte. Ujistěte se například, že adresáře obsahující knihovny produktu IBM MQ jsou v PATH.
CHYBA MQRC_CLASS_LIBRARY_ERROR (6001)	Jeden z volání produktu IBM MQ `classlibrary` vrátil neočekávanou hodnotu **ReasonCode** nebo **`CompletionCode`** . Podrobnosti naleznete v sestavě projevů prvního selhání. Vezměte na vědomí poslední použitou metodu/vlastnost a třídu a informujte IBM podporu daného problému.
MQRC_STRING_LENGTH_TOO_BIG (6002)	Byl proveden pokus o zápis řetězce formátu UTF s délkou větší než 65 535 bajtů do vyrovnávací paměti zpráv.
CHYBA MQRC_WRITE_VALUE_ERROR (6003)	Je použita hodnota, která je mimo rozsah; například `msg.WriteByte (240)`.
MQRC_PACKED_DECIMAL_ERROR (6004)	Byl proveden pokus o čtení zabaleného dekadického čísla z vyrovnávací paměti zpráv, ale data na datovém ukazateli nejsou v platném zhuštěném formátu dat.
MQRC_FLOAT_CONVERSION_ERROR (6005)	Byl proveden pokus o čtení jednoho nebo dvojnásobného čísla s pohyblivou řádovou čárkou z vyrovnávací paměti zpráv, ale data na ukazateli dat nejsou v náležitém formátu s plovoucí řádovou čárkou.

Kód příčiny	Vysvětlení
MQRC_REOPEN_EXCL_INPUT_ERROR (6100)	Otevřený objekt nemá správná nastavení **OpenOptions** a vyžaduje jednu nebo více dalších voleb. Je požadováno implicitní znovuotevření, ale uzavření bylo zabráněno, protože fronta je otevřená pro výlučný vstup a uzavření by prezentovalo okno příležitostí pro ostatní potenciálně získat přístup do fronty. Hodnoty parametru **OpenOptions** nastavte explicitně tak, aby pokrýly všechny možnosti tak, aby implicitní opětovné zahájení nebylo požadováno.
MQRC_REOPEN_INQUIRE_ERROR (6101)	Otevřený objekt nemá správná nastavení **OpenOptions** a vyžaduje jednu nebo více dalších voleb. Je požadováno implicitní znovuotevření, ale uzavření bylo zabráněno, protože jedna nebo více charakteristik objektu musí být zkontrolována dynamicky před uzavřením, a hodnoty **OpenOptions** již neobsahují volbu **MQOO_INQUIRE** . Hodnoty parametru **OpenOptions** nastavte explicitně tak, aby zahrnovaly volbu **MQOO_INQUIRE** .
MQRC_REOPEN_SAVED_CONTEXT_ERR (6102)	Otevřený objekt nemá správná nastavení **OpenOptions** a vyžaduje jednu nebo více dalších voleb. Je požadováno implicitní znovuotevření, ale uzavření bylo zabráněno, protože je fronta otevřena s volbou **MQOO_SAVE_ALL_CONTEXT** a destruktivní volání `Get` bylo provedeno dříve. To způsobilo, že uchovávané informace o stavu byly přidruženy k otevřené frontě a tyto informace by byly zničeny uzavřením. Hodnoty parametru **OpenOptions** nastavte explicitně tak, aby pokrýly všechny možnosti tak, aby implicitní opětovné zahájení nebylo požadováno.
MQRC_REOPEN_TEMPORARY_Q_ERROR (6103)	Otevřený objekt nemá správná nastavení **OpenOptions** a vyžaduje jednu nebo více dalších voleb. Je požadováno implicitní znovuotevření, ale uzavření bylo zabráněno, protože fronta je lokální frontou typu definice **MQQDT_TEMPORARY_DYNAMIC**, která by byla zničena uzavřením. Hodnoty parametru **OpenOptions** nastavte explicitně tak, aby pokrýly všechny možnosti tak, aby implicitní opětovné zahájení nebylo požadováno.
MQRC_ATTRIBUTE_LOCKED (6104)	Byl proveden pokus o změnu hodnoty nebo atributu objektu, když je tento objekt otevřený. Určité atributy, jako např. **AlternateUserId**, nelze změnit, když je objekt otevřený.
MQRC_CURSOR_NOT_VALID (6105)	Kurzor procházení pro otevřenou frontu byl zneplatněn, protože byl naposledy použit implicitní znovuotevření. Hodnoty parametru **OpenOptions** nastavte explicitně tak, aby pokrýly všechny možnosti tak, aby implicitní opětovné zahájení nebylo požadováno.

Kód příčiny	Vysvětlení
CHYBA MQRC_ENCODING_ERROR (6106)	Kódování další položky zprávy musí být **MQENC_NATIVE** kódování pro čtení.
CHYBA MQRC_STRUTCID_ERROR (6107)	Struktura ID další položky zprávy, která je odvozena od 4 znaků začínajících na ukazateli dat, buď chybí, nebo je nekonzistentní s typem proměnné, do níž je položka čtena.
MQRC_NULL_POINTER (6108)	Byl zadán ukazatel s hodnotou Null, je-li požadován nebo odvozený ukazatel s hodnotou jinou než Null. Tato situace může být způsobena použitím explicitních deklarací pro objekty produktu IBM MQ , které se používají z parametrů Visual Basic nebo Excel jako parametry volání. Příklad: • Z Visual Basic, `dim msg as Object` funguje správně, zatímco `dim msg as MqMessage` nemusí pracovat správně. • Z Visual Basic, s frontou definovanou a nastavenou, `dim msg as MqMessageq.put msg` pracuje správně, zatímco z aplikace Excel tento příkaz generuje výjimku `MQRC_NULL_POINTER` .
ODKAZ MQRC_NO_CONNECTION_REFERENCE (6109)	Objekt `MQQueue` ztratil spojení s objektem `MQQueueManager` . K tomu dojde, pokud je správce front odpojen. Odstraňte objekt `MQQueue` .
MQRC_NO_BUFFER (6110)	Není k dispozici žádná vyrovnávací paměť. Pro objekt `MQMessage` nelze vyrovnávací paměť přidělit, protože ve stavu objektu je vnitřní nekonzistence.
MQRC_BINARY_DATA_LENGTH_ERROR (6111), CHYBA	Délka binárních dat je nekonzistentní s délkou cílového atributu. Nula je správná délka pro všechny atributy. 24 je správná délka pro atribut **CorrelationId** a pro atribut **MessageId** . 32 je správná délka pro atribut **AccountingToken** .
MQRC_BUFFER_NOT_AUTOMATIC (6112)	Velikost uživatelsky definované a spravované vyrovnávací paměti nelze změnit. Vzhledem k tomu, že vyrovnávací paměť zpráv je spravována systémem, znamená to vnitřní nekonzistenci.
MQRC_INSUFFICIENT_BUFFER (6113)	Po ukazateli dat pro umístění požadavku není k dispozici dostatek prostoru vyrovnávací paměti. Důvodem může být skutečnost, že nelze změnit velikost vyrovnávací paměti.
NEDOSTATEČNÁ DATA MQRC_INSUFFICIENT_DATA (6114)	Po ukazateli dat pro uložení požadavku na čtení nejsou k dispozici dostatečná data. Zmenšete vyrovnávací paměť na správnou velikost a znovu si přečtěte data.

Kód příčiny	Vysvětlení
MQRC_DATA_OŘÍZNUTÁ (6115)	Data byla zkrácena při kopírování z jedné vyrovnávací paměti do jiné. Důvodem může být skutečnost, že nelze změnit velikost cílové vyrovnávací paměti, nebo protože se vyskytl problém s adresováním jedné nebo druhé vyrovnávací paměti, nebo protože vyrovnávací paměť je zmenšována s menší náhradou.
HODNOTA MQRC_ZERO_LENGTH (6116)	Byla dodána nulová délka, kde je kladná délka buď povinná, nebo implikovaná.
MQRC_NEGATIVNÍ_DÉLKA (6117)	Byla zadána záporná délka, kde je vyžadována nulová nebo kladná délka.
MQRC_NEGATIVNÍ_POSUN (6118)	Byl zadán záporný posun, u kterého je vyžadována nulová nebo kladná odchylka.
FORMÁT MQRC_INCONSISTENT_FORMAT (6119)	Formát další položky zprávy je nekonzistentní s typem proměnné, do níž je položka čtena.
MQRC_INCONSISTENT_OBJECT_STATE (6120)	Je zde nekonzistence mezi tímto objektem, který je otevřen, a odkazovaným objektem `MQQueueManager` , který není připojen.
MQRC_CONTEXT_OBJECT_NOT_VALID (6121)	Odkaz kontextu produktu `MQPutMessageOptions` neodkazuje na platný objekt `MQQueue` . Objekt byl již zničen.
CHYBA MQRC_CONTEXT_OPEN_ERROR (6122)	Odkaz na kontext produktu `MQPutMessageOptions` odkazuje na objekt `MQQueue` , který nelze otevřít pro vytvoření kontextu. Důvodem může být skutečnost, že objekt `MQQueue` má nevhodné otevřené volby. Zkontrolujte kód příčiny odkazovaného objektu, abyste zavedli příčinu.
CHYBA MQRC_STRUC_LENGTH_ERROR (6123)	Délka vnitřní datové struktury je nekonzistentní s jejím obsahem. Pro záhlaví `MQRMH` je délka nepostačující k tomu, aby obsahovala pevná pole a všechna data offsetu.
MQRC_NOT_CONNECTED (6124)	Metoda se nezdařila, protože požadované připojení ke správci front není k dispozici a nelze implicitně vytvořit připojení.
MQRC_NOT_OPEN (6125)	Metoda selhala, protože objekt IBM MQ není otevřen a otevření nemůže být implicitně provedeno.
MQRC_DISTRIBUTION_LIST_EMPTY (6126)	Nezdařilo se otevřít `MQDistributionList` , protože v rozdělovníku nejsou žádné objekty `MQDistributionListItem` . Opravná akce: Přidejte alespoň jeden objekt `MQDistributionListItem` do rozdělovníku.
MQRC_INCONSISTENT_OPEN_OPTIONS (6127)	Metoda selhala, protože objekt je otevřený a otevřené volby nejsou konzistentní s požadovanou operací. Nápravná akce: Otevřete objekt s vhodnými volbami otevření a zopakujte operaci.

Kód příčiny	Vysvětlení
MQRC_WRONG_VERSION (6128)	Metoda selhala, protože číslo verze zadané nebo zjištěné je buď chybné, nebo není podporované.

Nástroj na úrovni kódu

Možná vás bude požádat o servisní tým IBM , která úroveň kódu jste nainstalovali.

Chcete-li tuto skutečnost zjistit, spusťte obslužný program 'MQAXLEV'.

Z příkazového řádku přejděte do adresáře obsahujícího soubor MQAX200.dll nebo přidejte úplnou délku cesty a zadejte:

```
MQAXLev MQAX200.dll > MQAXLEV.OUT
```

kde MQAXLEV.OUT je název výstupního souboru.

Pokud neuvedete výstupní soubor, podrobnosti se zobrazí na obrazovce.

Příklad výstupního souboru z nástroje na úrovni kódu je podrobně popsán v následujícím příkladu:

Příklad výstupního souboru z nástroje na úrovni kódu

```
5639-B43 (C) Copyright IBM Corp. 1996, 2023. ALL RIGHTS RESERVED.
     ***** Code Level is 5.1 *****
lib/mqole/mqole.cpp, mqole, p000, p000 L981119       1.8 98/08/21
lib/mqlsx/gmqdyn0a.c, mqlsx, p000, p000 L990212      1.6 99/02/11 16:40:24
lib/mqlsx/pc/gmqdyn1p.c, mqlsx, p000, p000 L990212   1.6 99/02/11 16:44:14
lib/mqlsx/xmqcsa.c, mqole, p000, p000 L990216        1.3 99/02/15 13:24:34
lib/mqlsx/xmqfdca.c, mqlsx, p000, p000 L990212       1.3 99/02/11 16:40:35
lib/mqlsx/xmqtrca.c, mqlsx, p000, p000 L990212       1.5 99/02/11 16:12:02
lib/mqlsx/xmqutila.c, mqlsx, p000, p000 L990212      1.3 99/02/11 16:40:40
lib/mqlsx/xmqutl1a.c, mqlsx, p000, p000 L990212      1.4 99/02/11 16:40:30
lib/mqlsx/xmqcnv1a.c, mqlsx, p000, p000 L990212      1.9 99/02/11 16:40:56
lib/mqlsx/xmqmsg.c, mqole, p000, p000 L990219        1.11 99/02/18 12:12:59
```

Rozhraní ActiveX k rozhraní MQAI

Stručný přehled rozhraní COM a jejich použití v rozhraní MQAI najdete v tématu “Použití rozhraní Model objektu Component Model (IBM MQ Automation Classes for ActiveX)” na stránce 561.

Rozhraní MQAI umožňuje aplikacím vytvářet a odesílat příkazy PCF (Programmable Command Format) bez přímého získávání a formátování vyrovnávacích pamětí s proměnnou délkou požadovanou pro PCF. Další informace o rozhraní MQAI najdete v tématu Administrační rozhraní produktu IBM MQ (MQAI). Třída MQAI ActiveX MQBag zapouzdřuje datové balíky podporované rozhraním MQAI způsobem, který je možné použít v libovolném jazyce, který podporuje vytváření objektů COM; například Visual Basic, C + +, Javaa další skriptovací klienti ActiveX .

Rozhraní MQAI ActiveX je určen pro použití s třídami MQAX, které poskytují rozhraní COM pro rozhraní MQI. Další informace o třídách MQAX viz “Návrh aplikací MQAX, které přistupují k jiným aplikacím nežActiveX” na stránce 562.

Rozhraní ActiveX poskytuje jednu třídu s názvem MQBag. Tato třída se používá k vytváření datových pytlů MQAI a jejich vlastnosti a metody jsou použity k vytvoření a práci s datovými položkami v každém balíku. Metoda Execute MQBag odesílá data balíku do správce front produktu IBM MQ jako zprávu PCF a shromažďuje odpovědi.

Další informace o třídě MQBag, jejích vlastnostech a metodách viz “Třída MQBag” na stránce 642.

Zpráva PCF se odešle na uvedený objekt správce front, volitelně pomocí uvedených front požadavků a odpovědí. Odpovědi jsou vráceny v novém objektu MQBag. Úplná sada příkazů a odpovědí je popsána

v části Definice formátů Programovatelných příkazů. Příkazy lze odesílat libovolnému správci front v síti IBM MQ výběrem odpovídajících front požadavků a odpovědí.

Třída MQBag

Třída, MQBag, se používá k vytvoření objektů MQBag podle potřeby. Je-li vytvořena instance, třída MQBag vrátí nový odkaz na objekt MQBag.

Vytvořte objekt MQBag ve Visual Basic následujícím způsobem:

```
Dim mqbag As MQBag
Set mqbag = New MQBag
```

Vlastnost MQBag

Vlastnosti objektů MQBag jsou vysvětleny v následujícím seznamu:

- “Vlastnost položky” na stránce 643.
- “Vlastnost Count” na stránce 644.
- “Vlastnost Options” na stránce 644.

Metody MQBag

Metody objektů MQBag jsou vysvětleny v následujícím seznamu:

- “Přidat metodu” na stránce 645.
- “Metoda AddInquiry” na stránce 645.
- “Vymazat metodu” na stránce 646.
- “Provést metodu” na stránce 646.
- “Metoda FromMessage” na stránce 647.
- “Metoda ItemType” na stránce 647.
- “metoda odebrání” na stránce 648.
- “Metoda selektoru” na stránce 648.
- “Metoda ToMessage” na stránce 649.
- “Oříznout metodu” na stránce 649.

Zpracování chyb

Pokud je při operaci na objektu MQBag zjištěna chyba, včetně chyb, které byly vráceny do balíku podkladovým objektem MQAX nebo MQAI, dojde k výjimce chyby. Třída MQBag podporuje informační rozhraní COM ISupportError, takže následující informace jsou k dispozici pro rutinu ošetření chyb:

- Číslo chyby: složený z kódu příčiny IBM MQ pro zjištěnou chybu a kód zařízení COM. Pole zařízení, jako standard pro COM, označuje oblast odpovědnosti za chybu. Pro chyby zjištěné IBM MQ je to vždy FACILITY_ITF.
- Zdroj chyb: identifikuje typ a verzi objektu, který chybu zjistil. V případě chyb zjištěných během operací MQBag je tento zdroj chyb vždy MQBag.MQBag1.
- Popis chyby: řetězec udávající symbolický název pro kód příčiny IBM MQ .

Způsob přístupu k informacím o chybě závisí na jazyku skriptování. Například ve Visual Basicu se informace vrátí v objektu Err a kód příčiny IBM MQ se získá odečtením konstanty vbObjectz Err.Number.

ReasonCode = Err.Number -Chyba objektu vbObject

Pokud metoda Execute MQBag odešle zprávu PCF a odpověď je přijata, operace se považuje za úspěšnou, ačkoli odeslaný příkaz mohl selhal. V tomto případě obsahuje samotný vak pro odpověď kódy příčiny dokončení a chyby, jak je popsáno v tématu Definice formátů Programovatelných příkazů.

Vlastnost položky

Účel

Vlastnost Položka reprezentuje položku v balíku. Používá se k nastavení nebo dotazu na hodnotu položky. Použití této vlastnosti odpovídá následujícím voláním MQAI:

- "ŘetězecmqSet"
- "mqSetCelé číslo"
- "mqInquireCelé číslo"
- "ŘetězecmqInquire"
- "mqInquireBag"

v příručce Přehled formátů Programovatelných příkazů.

Formát

Položka (Selektor, ItemIndex, Hodnota)

Parametry

Selektor (VARIANT)-vstup
Selektor položky, která má být nastavena, nebo vyšetřován.

Při zjišťování informací o položce je výchozím nastavením hodnota MQSEL_ANY_USER_SELECTOR. Při nastavování položky je výchozím nastavením hodnota MQIA_LIST nebo MQCA_LIST.

Pokud `Selector` není typu long, hodnota MQRC_SELECTOR_TYPE_ERROR.

Tento parametr je volitelný.

ItemIndex (LONG)-vstup
Tato hodnota identifikuje výskyt položky uvedeného selektoru, který má být nastaven nebo se má provést zjišťování. MQIND_NONE je výchozí hodnota.

Tento parametr je volitelný.

Value (VARIANT)-input/output
Vrácená hodnota nebo hodnota, která má být nastavena. Při zjišťování informací o položce může být návratová hodnota typu long, string nebo MQBag. Když však nastavujete položku, musí být hodnota typu long nebo string; pokud tomu tak není, výsledky MQRC_ITEM_VALUE_ERROR.

Vyvolání jazyka Visual Basic

Při zjišťování informací o hodnotě položky v rámci balíku:

```
Value = mqbag[.Item]([Selector],
[ItemIndex])
```

Pro odkazy MQBag:

```
Set abag = mqbag[.Item]([Selector].
[ItemIndex])
```

Chcete-li nastavit hodnotu položky v balíku, postupujte takto:

```
mqbag[.Item]([Selector],
[ItemIndex]) = Value
```

Vlastnost Count

Účel

Vlastnost Count představuje počet datových položek v balíku. Tato vlastnost odpovídá volání MQAI, "mqCountItems," v příručce Přehled formátů Programovatelných příkazů.

Formát

Počet (*`Selector, Value`*)

Parametry

Selektor (VARIANT)-vstup
Selektor datových položek, které mají být zahrnuty do počtu.

MQSEL_ALL_USER_SELECTORS je výchozí hodnota.

Pokud `Selector` není typu long, je vrácen objekt MQRC_SELECTOR_TYPE_ERROR.

Hodnota (LONG)-výstup
Počet položek v balíku zahrnutých do *`Selector`*.

Vyvolání jazyka Visual Basic

Chcete-li vrátit počet položek v balíku:

```
ItemCount = mqbag.Count([Selector])
```

Vlastnost Options

Účel

Volby vlastnosti Volby nastavuje volby pro použití balíku. Tato vlastnost odpovídá parametru **`Options`** volání MQAI, "mqCreateBag," v příručce Přehled formátů Programovatelných příkazů.

Formát

Volby (*`Options`*)

Parametry

Volby (LONG)-vstupní/výstupní
Volby balíku.

Poznámka: Volby balíku musí být nastaveny ***před*** datovými položkami, aby byly přidány nebo nastaveny v rámci balíku. Pokud jsou volby změněny, když balík není prázdný, výsledky chyb MQRC_OPTIONS_ERROR. To platí i v případě, že se obal následně vymaže.

Vyvolání jazyka Visual Basic

Při zjišťování informací o volbách položky v rámci balíku:

```
Options = mqbag.Options
```

Chcete-li nastavit volbu položky v balíku, postupujte takto:

```
mqbag.Options = Options
```

Metody MQBag

Metody objektů MQBag jsou vysvětleny na následujících stránkách.

Přidat metodu

Účel

Metoda Add přidá datovou položku do balíku. Tato metoda odpovídá volání MQAI, "mqAddInteger" a "mqAddString," v příručce Přehled formátů Programovatelných příkazů.

Formát

Přidat (*`Value, Selector`*)

Parametry

Hodnota (VARIANT)-vstup
Celé číslo nebo řetězcová hodnota datové položky.

Selektor (VARIANT)-vstup
Selektor identifikující položku, která má být přidána.

V závislosti na typu `Value`je výchozí hodnota MQIA_LIST nebo MQCA_LIST. Pokud parametr **`Selector`** není typu long, výsledky MQRC_SELECTOR_TYPE_ERROR.

Vyvolání jazyka Visual Basic

Chcete-li přidat položku do balíku:

```
mqbag.Add(Value,[Selector])
```

Metoda AddInquiry

Účel

Metoda AddInquiry přidá selektor uvádějící atribut, který se má vrátit, když se odešle balík administrace k provedení příkazu INQUIRE. Tato metoda odpovídá volání MQAI, "mqAddInquiry,", v produktu Přehled formátů Programovatelných příkazů.

Formát

AddInquiry *`(Inquiry`*)

Parametry

Dotazová (LONG)-vstup
Selektor atributu IBM MQ , který má být vrácen příkazem administrace INQUIRE.

Vyvolání jazyka Visual Basic

Chcete-li použít metodu AddInquiry :

```
mqbag.AddInquiry(Inquiry)
```

Vymazat metodu

Účel

Metoda Clear odstraní všechny datové položky z balíku. Tato metoda odpovídá volání MQAI, "mqClearBag,", v příručce Přehled formátů Programovatelných příkazů.

Formát

Vymazat

Vyvolání jazyka Visual Basic

Chcete-li odstranit všechna data itmes z balíku:

```
mqbag.Clear
```

Provést metodu

Účel

Metoda Execute odešle na příkazový server zprávu příkazu administrace a čeká na všechny zprávy odpovědí. Tato metoda odpovídá volání MQAI, "mqExecute,", v rámci Přehled formátů Programovatelných příkazů.

Formát

Spuštění produktu *(QueueManager, Command, OptionsBag, RequestQ, ReplyQ, ReplyBag*)

Parametry

QueueManager (MQQueueManager)-vstup
Správce front, ke kterému je aplikace připojena.

Příkaz (LONG)-vstup
Příkaz, který má být proveden.

OptionsBag (MQBag)-vstup
Sáček obsahující volby, které ovlivňují zpracování volání.

RequestQ (MQQueue)-vstup
Fronta, do které bude umístěna zpráva příkazu administrace.

ReplyQ (MQQueue)-vstup
Fronta, na které jsou přijímány všechny zprávy odpovědi.

ReplyBag (MQBag)-výstup
Odkaz na balík obsahující data ze zpráv odpovědi.

Vyvolání jazyka Visual Basic

Chcete-li odeslat zprávu s příkazovým příkazem a čekat na všechny zprávy odpovědí, postupujte takto:

```
Set ReplyBag = mqbag.Execute(QueueManager, Command,
[OptionsBag],[RequestQ],[ReplyQ])
```

Metoda FromMessage

Účel

Metoda FromMessage načte data ze zprávy do tašky. Tato metoda odpovídá volání MQAI, "mqBufferToBag," v rámci Přehled formátů Programovatelných příkazů.

Formát

FromMessage *(Message, OptionsBag)*

Parametry

Zpráva (MQMessage)-vstup
Zpráva obsahující data, která mají být převedena.

OptionsBag (MQBag)-vstup
Volby pro řízení zpracování volání.

Vyvolání jazyka Visual Basic

Chcete-li načíst data ze zprávy do balíku, postupujte takto:

```
mqbag.FromMessage(Message,[OptionsBag])
```

Metoda ItemType

Účel

Metoda ItemType vrací typ hodnoty v zadané položce v balíku. Tato metoda odpovídá volání MQAI, "mqInquireItemInfo," v rámci Přehled formátů Programovatelných příkazů.

Formát

ItemType *(Selector, ItemIndex, ItemType)*

Parametry

Selektor (VARIANT)-vstup
Selektor identifikující položku, která má být dotazovaná.

MQSEL_ANY_USER_SELECTOR je výchozí hodnota. Pokud parametr **Selector** není typu long, výsledky MQRC_SELECTOR_TYPE_ERROR.

ItemIndex (LONG)-vstup
Index položek, které mají být dotazovány.

MQIND_NONE je výchozí hodnota.

ItemType (LONG)-výstup
Datový typ zadané položky.

Poznámka: Musí být zadán buď parametr **Selector** , **ItemIndex** nebo oba parametry. Není-li zadán žádný parametr, budou výsledkem operace MQRC_PARAMETER_MISSING.

Vyvolání jazyka Visual Basic

Chcete-li vrátit typ hodnoty, postupujte takto:

```
ItemType = mqbag.ItemType([Selector],
[ItemIndex])
```

metoda odebrání

Účel

Metoda Odebrat odstraní položku z balíku. Tato metoda odpovídá volání MQAI, "mqDeleteItem," v příručce Přehled formátů Programovatelných příkazů.

Formát

Odebrat *(Selector, ItemIndex)*

Parametry

Selektor (VARIANT)-vstup
Selektor identifikující položku, která má být odstraněna.

MQSEL_ANY_USER_SELECTOR je výchozí hodnota. Pokud parametr **Selector** není typu long, výsledky MQRC_SELECTOR_TYPE_ERROR.

ItemIndex (LONG)-vstup
Index položky, která má být odstraněna.

MQIND_NONE je výchozí hodnota.

Poznámka: Musí být zadán buď parametr **Selector** , **ItemIndex** nebo oba parametry. Není-li zadán žádný parametr, budou výsledkem operace MQRC_PARAMETER_MISSING.

Vyvolání jazyka Visual Basic

Chcete-li odstranit položku z balíku:

```
mqbag.Remove([Selector],[ItemIndex])
```

Metoda selektoru

Účel

Metoda Selector vrací selektor zadané položky v rámci balíku. Tato metoda odpovídá volání MQAI, "mqInquireItemInfo," v rámci Přehled formátů Programovatelných příkazů.

Formát

Selektor *(Selector, ItemIndex, OutSelector)*

Parametry

Selektor (VARIANT)-vstup
Selektor identifikující položku, která má být dotazovaná.

MQSEL_ANY_USER_SELECTOR je výchozí hodnota. Pokud parametr **Selector** není typu long, výsledky MQRC_SELECTOR_TYPE_ERROR.

ItemIndex (LONG)-vstup
Index položky, která má být dotazovaná.

MQIND_NONE je výchozí hodnota.

OutSelector (VARIANT)-výstup
Selektor zadané položky.

Poznámka: Musí být zadán buď parametr **Selector** , **ItemIndex** nebo oba parametry. Není-li zadán žádný parametr, budou výsledkem operace MQRC_PARAMETER_MISSING.

Vyvolání jazyka Visual Basic

Chcete-li vrátit selektor položky, postupujte takto:

```
OutSelector = mqbag.Selector([Selector],
[ItemIndex])
```

Metoda ToMessage

Účel

Metoda ToMessage vrací odkaz na objekt MQMessage. Odkaz obsahuje data z balíku. Tato metoda odpovídá volání MQAI, "mqBagToBuffer," v publikaci Přehled formátů Programovatelných příkazů.

Formát

ToMessage *(OptionsBag, Message)*

Parametry

OptionsBag (MQBag)-vstup
Sáček obsahující volby, které řídí zpracování metody.

Výstup zprávy (MQMessage)-výstup
Odkaz na objekt MQMessage obsahující data z balíku.

Vyvolání jazyka Visual Basic

Chcete-li použít metodu ToMessage , postupujte takto:

```
Set Message = mqbag.ToMessage([OptionsBag])
```

Oříznout metodu

Účel

Metoda Truncate snižuje počet uživatelských položek v balíku. Tato metoda odpovídá volání MQAI, "mqTruncateBag," v příručce Přehled formátů Programovatelných příkazů.

Formát

Oseknout *(ItemCount)*

Parametry

ItemCount (LONG)-vstup
Počet položek uživatele, které mají zůstat v balíku po oříznutí.

Vyvolání jazyka Visual Basic

Chcete-li snížit počet uživatelských položek v balíku, postupujte takto:

```
mqbag.Truncate(ItemCount)
```

O produktu IBM MQ Automation Classes for ActiveX Starter samples

Tato příloha popisuje ukázky produktu IBM MQ Automation Classes for ActiveX Starter a vysvětluje, jak je používat.

Produkt IBM MQ for Windows poskytuje následující ukázkové programy Visual Basic:

- MQAXTRIV.VBP
- MQAXBSRV.VBP
- MQAXDLST.VBP
- MQAXCLSS.VBP

Tyto ukázky jsou spuštěny ve Visual Basic 4 nebo Visual Basic 5. Naleznete je v adresáři ... \tools\mqax\samples\vb.

Ve stejném adresáři také najdete ukázky pro Microsoft Excel a html. Patří mezi ně:

- MQAX.XLS
- MQAXTRIV.XLS
- MQAXTRIV.HTM

Poznámka: Používáte-li Visual Basic 5, **musíte** vybrat a instalovat komponentu Visual Basic grid32.ocx.

Co je prokázáno ve vzorcích

Ukázky ukazují, jak používat třídy automatizace IBM MQ pro ActiveX k:

- Připojit se ke správci front
- Přístup k frontě
- Vložit zprávu do fronty
- Získat zprávu z fronty

Centrální část vzorku Visual Basic se zobrazí na následujících stránkách.

“Příprava na spuštění ukázek” na stránce 651 a

“Ošetření chyb ve vzorcích” na stránce 651

Spuštění ukázek ActiveX Starter

Před spuštěním ukázkových tříd produktu IBM MQ pro ukázky ActiveX zkontrolujte, že je spuštěn výchozí správce front a že jste vytvořili požadované definice front. Podrobnosti o vytvoření a spuštění správce front a vytvoření fronty naleznete v tématu Správa. Ukázka používá frontu SYSTEM.DEFAULT.LOCAL.QUEUE , která by měla být definována na každém normálně nastaveném serveru IBM MQ .

Různé způsoby použití datových pytlů jsou uvedeny v následujícím seznamu:

- Připojit se ke správci front
- Přístup k frontě
- Vložit zprávu do fronty
- Získat zprávu z fronty

Informace o ukázkách spouštěče MQAX pro produkt Microsoft Basic 4 nebo novější naleznete v tématu “Spuštění ukázky MQAXTRIV” na stránce 651 .

Informace o ukázce, která umožňují procházet vlastnosti a metody správců front a objektů front, naleznete v tématu “Spuštění ukázky MQAXCLSS” na stránce 653 .

Informace o ukázce MQAXDLST obsahuje “Ukázka MQAXDLST” na stránce 653

Informace o spuštění ukázky spouštěče MQAX pro produkt Microsoft Excel 95 nebo vyšší verze MQAXTRIV.XLS, viz “Spuštění příkazu MQAXTRIV.XLS” na stránce 653.

Informace o spuštění ukázkové aplikace Bank s produktem MQAX.XLS, viz “Spuštění ukázkové aplikace Bank s parametrem MQAX.XLS” na stránce 653

Informace o startovní ukázce pomocí kompatibilního webového prohlížeče ActiveX viz “Ukázkový příklad použití kompatibilního webového prohlížeče ActiveX” na stránce 654

Příprava na spuštění ukázek

Chcete-li spustit některý z ukázek, je třeba provést jednu z následujících možností v závislosti na tom, které ukázky hodláte spustit.

- Microsoft Visual Basic 4 (nebo novější)
- Microsoft Excel 95 (nebo novější)
- Webový prohlížeč

Potřebujete také:

- Správce front produktu IBM MQ je spuštěn.
- Fronta IBM MQ je již definována.

Ošetření chyb ve vzorcích

Většina ukázek poskytnutých v balíku IBM MQ Automation Classes for ActiveX vykazuje malou nebo žádnou chybu při zpracování chyb. Další informace o ošetření chyb viz “Ošetření chyb” na stránce 566.

Spuštění ukázky MQAXTRIV

1. Spusťte správce front.
2. V produktu Windows Explorer nebo File Managervyberte ikonu ukázky MQAXTRIV.VBP (soubor projektu Visual Basic) a otevřete soubor.

   Spustí se program Visual Basic a otevře se soubor MQAXTRIV.VBP.
3. V Visual Basic stiskněte funkční klávesu 5 (F5) pro spuštění ukázky.
4. Klepněte kamkoli do formuláře okna, **MQAX trivial tester**.

Pokud vše funguje správně, pozadí okna by se mělo změnit na zelenou. Pokud se vyskytl problém s nastavením, pozadí okna by se mělo změnit na červené a chybové informace se zobrazí.

Následující obrázek ukazuje centrální část ukázky jazyka Visual Basic.

```
Option Explicit

Private Sub Form_Click()

'*************************************************************************
'* This simple example illustrates how to put and get an IBM MQ message to
'* and from an IBM MQ message queue. The data from the message returned by the
'* get is read and compared with that from the original message.
'*************************************************************************
Dim MQSess As MQSession                 '* session object
Dim QMgr As MQQueueManager              '* queue manager object
Dim Queue As MQQueue                    '* queue object
```

```
Dim PutMsg As MQMessage                 '* message object for put
Dim GetMsg As MQMessage                 '* message object for get
Dim PutOptions As MQPutMessageOptions   '* put message options
Dim GetOptions As MQGetMessageOptions   '* get message options
Dim PutMsgStr As String                 '* put message data string
Dim GetMsgStr As String                 '* get message data string
'*************************************************************************
'* Handle errors
'*************************************************************************
On Error GoTo HandleError

'*************************************************************************
'* Initialize the current position for the form
'*************************************************************************
CurrentX = 0
CurrentY = 0

'*************************************************************************
'* Create the MQSession object and access the MQQueueManager and (local) MQQueue
'*************************************************************************
Set MQSess = New MQSession
Set QMgr = MQSess.AccessQueueManager("")
Set Queue = QMgr.AccessQueue("SYSTEM.DEFAULT.LOCAL.QUEUE", _
                      MQOO_OUTPUT Or MQOO_INPUT_AS_Q_DEF)

'*************************************************************************
'* Create a new MQMessage object for use with put, add some data then create an
'* MQPutMessageOptions object and put the message
'*************************************************************************
Set PutMsg = MQSess.AccessMessage()
PutMsgStr = "12345678 " & Time
PutMsg.MessageData = PutMsgStr
Set PutOptions = MQSess.AccessPutMessageOptions()
Queue.Put PutMsg, PutOptions

'*************************************************************************
'* Create a new MQMessage object for use with get, set the MessageId (to that of
'* the message that was put), create an MQGetMessageOptions object and get the
'* message.
'*
'* Note: Setting the MessageId ensures that the get returns the MQMessage
'* that was put earlier.
'*************************************************************************

Set GetMsg = MQSess.AccessMessage()
GetMsg.MessageId = PutMsg.MessageId
Set GetOptions = MQSess.AccessGetMessageOptions()
Queue.Get GetMsg, GetOptions
'*************************************************************************
'* Read the data from the message returned by the get, compare it with
'* that from the original message and output a suitable message.
'*************************************************************************
GetMsgStr = GetMsg.MessageData
Cls
If GetMsgStr = PutMsgStr Then
    BackColor = RGB(127, 255, 127) '* set to green for ok
    Print
    Print "Message data comparison was successful."
    Print "Message data: """ & GetMsgStr & """"

Else
    BackColor = RGB(255, 255, 127)  '* set to amber for compare error
    Print "Compare error: "
    Print "The message data returned by the get did not match the " &_
    "input data from the original message that was put."
    Print
    Print "Input message data:        """ & PutMsgStr & """"
    Print "Returned message data: """ & GetMsgStr & """"
End If

Exit Sub
'*************************************************************************
'* Handle errors
'*************************************************************************
HandleError:
Dim ErrMsg As String
Dim StrPos As Integer

Cls
BackColor = RGB(255, 0, 0)  '* set to red for error
Print "An error occurred as follows:"
Print ""
```

```
If MQSess.CompletionCode <> MQCC_OK Then
    ErrMsg = Err.Description
    StrPos = InStr(ErrMsg, " ")            '* search for first blank
    If StrPos > 0 Then
        Print Left(ErrMsg, StrPos)         '* print offending MQAX object name
    Else
        Print Error(Err)                   '* print complete error object
    End If
    Print ""
    Print "IBM MQ Completion Code = " & MQSess.CompletionCode
    Print "IBM MQ Reason Code = " & MQSess.ReasonCode
    Print "(" & MQSess.ReasonName & ")"
Else
    Print "Visual Basic error: " & Err
    Print Error(Err)
End If

Exit Sub

End Sub
```

Spuštění ukázky MQAXCLSS

Tato ukázka vám umožňuje procházet vlastnosti a metody správců front a objektů front.

1. Spusťte správce front.
2. Otevřete soubor MQAXCLSS.VBP, poklepáním na ikonu dokumentu v Průzkumníku Windows nebo klepnutím na Soubor-Otevřít z nabídky Soubor ve Visual Basicu.
3. Spusťte ukázku.
4. Zadejte příslušné správce front a názvy front a poté klepněte na příslušná tlačítka.

Ukázka MQAXDLST

Ukázka kódu Visual Basic MQAXDLST demonstruje použití distribučního seznamu k odeslání stejné zprávy do dvou front s jedním vkládanou. Chcete-li ukázku spustit, proveďte to samé jako pro ukázku MQAXCLSS.

Ukázka MQAX Starter pro produkt Microsoft Excel 95 nebo novější

Tento oddíl vysvětluje, jak spustit ukázku spouštěče MQAX pro produkt Microsoft Excel 95 nebo novější, MQAXTRIV.XLS.

Spuštění příkazu MQAXTRIV.XLS

1. Spusťte správce front.
2. V Průzkumníku nebo File Managervyberte ikonu pro vzorek MQAX MQAXTRIV.XLS.
3. Klepněte na tlačítko v sešitu.
4. Obrazovka je aktualizována zprávou o úspěchu (nebo selhání).

Spuštění ukázkové aplikace Bank s parametrem MQAX.XLS

Chcete-li spustit demonstraci Bank, postupujte podle následujících kroků.

1. Spusťte správce front.
2. Spusťte příkazový soubor IBM MQ MQSC, BANK.TST. Tím se nastaví potřebné definice front produktu IBM MQ .

   Chcete-li zjistit, jak používat příkazový soubor MQSC, prostudujte si téma Příkazy skriptu MQSC (Script).
3. Spusťte příkaz MQAXBSRV.VBP. Tento ukázkový program je server simulující back-endovou aplikaci a musí být spuštěn s produktem Microsoft Excel.
4. Spusťte MQAX.XLS. Tato ukázka je demonstrace klienta IBM MQ .
5. Vyberte zákazníka ze seznamu.

6. Klepněte na tlačítko **Odeslat**.

Po krátké přestávce (přibližně 3 sekundy) se pole naplní hodnotami a zobrazí se sloupcový graf.

Ukázkový příklad použití kompatibilního webového prohlížeče ActiveX

Poznámka: Chcete-li spustit tuto ukázku, musíte spustit webový prohlížeč kompatibilní s ActiveX . Produkt Microsoft Internet Explorer (ale ne Netscape Navigator) je kompatibilní webový prohlížeč.

Spuštění ukázky HTML

Tato ukázka demonstruje, jak lze vyvolat MQAX z jazyků VBScript i z produktu JavaScript.

1. Spusťte správce front.
2. Otevřete soubor, "MQAXTRIV.HTM", ve webovém prohlížeči kompatibilním s ActiveX .

   Můžete to provést buď dvojitým klepnutím na ikonu souboru v Průzkumníku Windows , nebo můžete zvolit Soubor-Otevřít z nabídky Soubor ve webovém prohlížeči kompatibilního s ActiveX .
3. Postupujte podle pokynů na obrazovce.

ULW V 9.0.0 Vyvíjení klientských aplikací AMQP

Podpora produktu IBM MQ pro rozhraní API AMQP, včetně rozhraní API produktu MQ Light , umožňuje administrátorovi produktu IBM MQ vytvořit kanál AMQP. Je-li tento kanál spuštěn, definuje číslo portu, které přijímá připojení z klientských aplikací AMQP.

Kanál AMQP lze instalovat v systémech UNIX, Linuxnebo Windows; není k dispozici v produktu IBM i nebo z/OS.

Rozhraní API produktu MQ Light je založeno na protokolu Oasis AMQP 1.0 . Pro produkt Node.js, Java, Ruby a Pythonjsou k dispozici rozhraní API systému zpráv.

Aplikace vyvinutá pro použití rozhraní API produktu MQ Light může být připojena buď k běhovému prostředí produktu MQ Light , ke správci front IBM MQ s kanálem AMQP, nebo jako instance služby MQ Light v produktu IBM Cloud (formerly Bluemix).

Vývoj klientů AMQP

Rozhraní API produktu MQ Light má za cíl usnadnit práci s prototypem a rychle vyvíjet obchodní aplikace. K dispozici jsou MQ Light API pro Node.js, Java, Ruby a Python, které jsou k dispozici na adrese https://github.com/mqlight.

Stažení ukázkových klientů AMQP

Produkt IBM MQ nedodává klienty MQ Light , ale můžete stáhnout a nainstalovat následující klienty MQ Light :

Node.js
Nainstalujte produkt MQ Light Node.js API do svého pracovního adresáře pomocí npm: `npm install mqlight@1.0`

Java
Stáhněte balík programu mqlight-distribuční balík pro požadovanou verzi z nástroje Maven Central a extrahujte jeho obsah. Dostupné verze balíků programu mqlight-distribuční balíky naleznete na webu Maven Central.

Ruby
Nainstalujte produkt MQ Light Ruby API do svého pracovního adresáře pomocí gem: `gem install mqlight --pre`

Python
Nainstalujte produkt MQ Light Python API do svého pracovního adresáře pomocí p-ip: `pip install mqlight --pre`

Ke stažení klienta produktu MQ Light patří několik ukázek, které demonstrují různé funkce zasílání zpráv:

- Odeslat ukázku
- Přijmout ukázku
- Ukázka UI Workout

Můžete si také stáhnout další open-source klienty AMQP založené na knihovnách Qpid Apache , další informace viz https://qpid.apache.org/index.html

Zabezpečení klientů AMQP

Informace o zabezpečení aplikací produktu MQ Light naleznete v tématu Zabezpečení klientů AMQP.

Implementace klientů AMQP do produktu IBM MQ

Je-li aplikace připravena k implementaci, vyžaduje všechny funkce monitorování, spolehlivosti a zabezpečení ostatních podnikových aplikací. Může si také vyměňovat data s ostatními podnikovými aplikacemi. Aplikace produktu MQ Light můžete implementovat do správce front produktu IBM MQ . Viz “Implementace aplikací produktu MQ Light do prostředí produktu IBM MQ s lokální verzí” na stránce 669 .

Pokud jste implementovali klienta AMQP, můžete si vyměňovat zprávy s aplikacemi produktu IBM MQ . Pokud například použijete klienta MQ Light Node.js k odeslání řetězcové zprávy produktu JavaScript , obdrží aplikace IBM MQ zprávu MQ , kde je pole formátu MQMD nastaveno na hodnotu MQSTR.

Správa kanálu AMQP

Kanál AMQP může být spravován stejným způsobem jako jiné kanály produktu MQ . K definování, spuštění, zastavení a správě kanálů můžete použít příkazy MQSC, zprávy příkazu PCF nebo IBM MQ Explorer . V části Vytvoření a použití kanálů AMQPjsou k dispozici ukázkové příkazy pro definování a spuštění připojení klientů ke správci front.

Je-li kanál AMQP spuštěn, můžete jej otestovat připojením aplikace MQ Light pomocí některé z následujících metod:

- Pomocí klienta IBM MQ Light pro Node.js a Java.
- Použití klienta IBM MQ Light pro Ruby a Python.
- Použití jiného klienta AMQP 1.0 . Příklad: Apache Qpid Proton.

Související informace
Vytvoření a použití kanálů AMQP
Zabezpečení klientů AMQP

ULW MQ Light a AMQP (Advanced Message Queuing Protocol)

Rozhraní API IBM MQ Light je založeno na protokolu spoje OASIS Standard AMQP 1.0 . AMQP uvádí, jak jsou zprávy odesílány mezi odesilateli a příjemci. Aplikace se chová jako odesilatel, když aplikace odesílá zprávu zprostředkovateli zpráv, jako například IBM MQ. Produkt IBM MQ vystupuje jako odesilatel při odesílání zprávy do aplikace AMQP.

Některé z výhod AMQP jsou následující:

- Otevřený standardizovaný protokol
- Kompatibilita s ostatními klienty s otevřeným zdrojem AMQP 1.0
- K dispozici řada implementací klientů s otevřeným zdrojovým kódem

Ačkoli se klient AMQP 1.0 může připojit ke kanálu AMQP, některé funkce AMQP nejsou podporovány, například transakce nebo více relací.

Další informace naleznete na webových stránkách AMQP.org a OASIS Standard AMQP 1.0 PDF.

Rozhraní API systému zpráv produktu MQ Light je založeno na AMQP 1.0. Rozhraní API poskytuje většinu schopnosti systému zpráv potřebné pro většinu toků zpráv typu publikování/odběr a systému zpráv typu point-to-point.

Rozhraní API produktu MQ Light má následující funkce posílání zpráv:

- Doručení zpráv s nejvyšší jednou zprávou
- Dodávka alespoň jednou.
- Adresování cíle řetězce tématu
- Životnost zprávy a místa určení
- Sdílené cíle, které umožňují více odběratelům sdílet pracovní zátěž
- Převzetí klienta pro snadné vyřešení zablokovaných klientů
- Konfigurovatelné čtení napřed zpráv
- Konfigurovatelné potvrzení zpráv

Kompletní dokumentace k rozhraní API produktu MQ Light naleznete na následujících webových serverech:

- Dokumentaci k rozhraní API Node.js viz https://www.npmjs.org/package/mqlight
- Dokumentaci k rozhraní API Ruby viz https://www.rubydoc.info/github/mqlight/ruby-mqlight
- Dokumentaci k rozhraní API Python naleznete v tématu https://python-mqlight.readthedocs.org .
- Dokumentaci k rozhraní API produktu Java naleznete v tématu https://mqlight.github.io/java-mqlight

Související informace

Vytvoření a použití kanálů AMQP

Zabezpečení klientů AMQP

ULW

Podpora AMQP 1.0

Kanály AMQP poskytují úroveň podpory pro aplikace AMQP 1.0-compliant .

Kanály AMQP podporují dílčí sadu protokolu AMQP 1.0 . Ke kanálu AMQP IBM MQ můžete připojit buď klienty MQ Light , nebo jiné kompatibilní klienty AMQP 1.0 . Chcete-li používat všechny funkce zasílání zpráv podporované kanály AMQP, musíte správně nastavit hodnotu některých polí AMQP 1.0 .

Tyto informační osnovy popisují způsob, jakým musí být pole AMQP formátována, a uvádí funkce specifikace AMQP 1.0 , které nejsou podporovány kanály AMQP.

Následující funkce specifikace AMQP 1.0 jsou buď nepodporovány, nebo jsou omezeny jejich použití:

Názvy odkazů

Kanály AMQP očekávají, že se název propojení AMQP bude řídit jedním ze tří formátů:

- Prostý téma (pro publikování a odběr)
  - Publikování zpráv: prostý řetězec tématu (například název odkazu na "`/sports/football`") způsobí publikování zprávy na téma `/sports/football` .
  - Přihlášení k odběru tématu pro příjem zpráv: Prostý řetězec tématu (například název odkazu "`/sports/football`" způsobí, že odběr bude definován v tématu `/sports/football` .
- Soukromé téma s komentářem (pro přihlášení k odběru)
  - Podrobný řetězec tématu, který popisuje soukromý odběr ve tvaru: "`private:`*`topic string`*" (například: "`private:/sports/football`"). Chování je identické s řetězcem s prostým tématem.

Deklarace `private` rozlišuje odběr specifický pro konkrétní klienta AMQP z odběru sdíleného mezi klienty.

- Sdílené téma s komentářem (pro přihlášení k odběru)
  - Podrobný řetězec tématu, který popisuje sdílený odběr ve tvaru: "`share:`*`share name`*`:`*`topic string`*" (například: "`share:bbc:/sports/football`").

Další informace o způsobu mapování zpráv AMQP na a ze zpráv produktu IBM MQ naleznete v tématu Mapování polí AMQP na pole produktu IBM MQ (příchozí zprávy).

Maximální délky pro řetězce témat, názvy sdílení a ID klientů

Řetězec tématu, sdílený název a ID klienta musí být obsaženy v 10237 bajtech. Kromě toho je maximální délka ID klienta 256 znaků.

Tyto maximální délky znamenají, že můžete mít jednu z následujících možností:

- velmi dlouhý řetězec tématu za předpokladu, že název sdílené složky je krátký.
- dlouhý název sdílení, ale krátký řetězec tématu

ID kontejnerů

Kanály AMQP očekávají, že se ID kontejneru AMQP Open performtive bude obsahovat jedinečné ID klienta MQ Light . Maximální délka ID klienta MQ Light je 256 znaků a ID může obsahovat alfanumerické znaky, znak procent (%), lomítko (/), tečku (.) a podtržítko (_).

Relace

Kanály AMQP podporují pouze jednu relaci AMQP. Klient AMQP, který se pokusí vytvořit více než jednu relaci AMQP, obdrží chybovou zprávu a je odpojen od kanálu.

Transakce

Kanály AMQP nepodporují transakce AMQP. Rámec pro připojení AMQP, který se pokusí koordinovat novou transakci nebo přenosový rámec AMQP, který se pokusí deklarovat novou transakci, bude odmítnut s chybovou zprávou.

Stav doručení

Kanály AMQP podporují pouze doručovací stav pro rámce přijetí typu Přijato.

Související informace

Vytvoření a použití kanálů AMQP

Zabezpečení klientů AMQP

ULW V 9.0.0 Mapování polí zpráv AMQP a IBM MQ

Zprávy AMQP se skládají ze záhlaví, doručovacích anotací, anotací zpráv, vlastností, vlastností aplikace, těla a zápatí.

Zprávy AMQP se skládají z následujících částí:

Header

Volitelné záhlaví obsahuje pět pevných atributů zprávy:

- **trvanlivý** -uvádí požadavky na životnost.
- **priority** -Relativní priorita zprávy
- **ttl** -doba života v milisekundách
- **first-nabyvatel** -Je-li to pravda, zpráva nebyla získána žádným jiným odkazem
- **delivery-count** -počet předchozích neúspěšných pokusů o doručení.

Doručení-anotace
Volitelné. Určuje nestandardní atributy záhlaví zprávy pro různé určené cílové skupiny. Anotace doručení předávají informace z odesílajícího partnera na příjemce typu peer příjmu.

Anotace-zprávy
Volitelné. Určuje nestandardní atributy záhlaví zprávy pro různé určené cílové skupiny. Sekce anotace zpráv se používá pro vlastnosti zprávy, které jsou zaměřeny na infrastrukturu a měly by být šířeny v každém kroku doručení.

Vlastnosti
Volitelné. Tato část je ekvivalentem deskriptoru zpráv MQ . Obsahuje následující pevná pole:

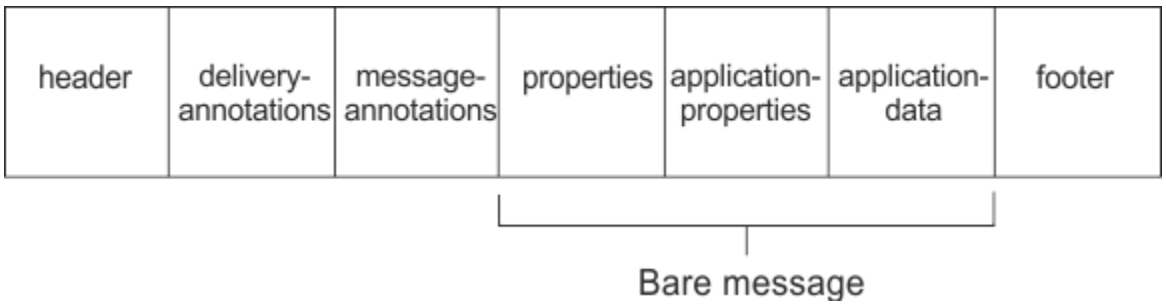
- **message-id** -Identifikátor zprávy aplikace
- **user-id** -ID uživatele vytvářející uživatele
- **to** -adresa uzlu, pro který je zpráva určena
- **subject** -předmět zprávy
- **reply-to** -uzel, na který je odeslána odpověď
- **correlation-id** -Identifikátor korelace aplikace.
- **content-type** -typ obsahu MIME
- **content-encoding** -Typ obsahu MIME. Používá se jako modifikátor typu content-type.
- **absolute-expiry-time** -doba, kdy je tato zpráva považována za ukončenou.
- **creation-time** -Čas, kdy byla tato zpráva vytvořena
- **group-id** -Skupina, do níž tato zpráva patří.
- **group-sequence** -pořadové číslo této zprávy v rámci příslušné skupiny.
- **reply-to-group-id** -Skupina, do níž náleží zpráva odpovědi.

Aplikace-vlastnosti
Ekvivalentní k vlastnostem zprávy produktu MQ .

Tělo
Ekvivalentní k informačním obsahu uživatele produktu MQ .

Zápatí
Volitelné. Zápatí se používá pro podrobnosti o zprávě nebo doručení, které lze vypočítat nebo vyhodnotit teprve po sestavení nebo zobrazení celé obnažené zprávy (například hašování zpráv, HMACs, podpisy a informace o šifrování).

Formát zprávy AMQP je znázorněn na následujícím obrázku:



Vlastnosti, vlastnosti aplikace a část aplikačních dat jsou známy jako "vlastní zpráva". Jedná se o zprávu odesílanou odesílatelem a je neměnná. Příjemce vidí celou zprávu, včetně záhlaví, zápatí, doručovacích anotací a anotací zpráv.

Úplný popis formátu zprávy AMQP 1.0 naleznete ve standardu OASIS na adrese https://docs.oasis-open.org/amqp/core/v1.0/amqp-core-complete-v1.0.pdf.

Související informace
Vytvoření a použití kanálů AMQP
Zabezpečení klientů AMQP

ULW V 9.0.0 Mapování polí IBM MQ na pole AMQP (odchozí zprávy)

Je-li publikována zpráva IBM MQ a produkt IBM MQ ji odešle do odběratele AMQP, rozšíří některé atributy zprávy IBM MQ do ekvivalentních atributů zprávy AMQP.

záhlaví

Záhlaví je zahrnuto pouze v případě, že jedno z pěti polí v záhlaví obsahuje jinou než výchozí hodnotu. Do záhlaví jsou zahrnuta pouze pole s nevýchozí hodnotou. Pět polí záhlaví se zpočátku odvozuje z ekvivalentní vlastnosti `mq_amqp.Hdr` , pokud je nastavena, a pak upravena tak, jak je zobrazeno v následující tabulce:

Tabulka 80. Mapování polí záhlaví

Pole	Výchozí hodnota	Hodnota
Trvalý	ne	Je splněn, pokud je parametr `MQMD.Persistence` nastaven na hodnotu `MQPER_PERSISTENT`, jinak hodnota false.
priority (priorita)	4	Od `mq_amqp.Hdr.Pri`, pokud je nastaven, nebo jinak z `MQMD.Priority`, pokud je nastaven. Není-li nastaven ani nastaven, nastavte na 4.
ttl	Není k dispozici.	`MQMD.Expiry` v milisekundách. Je-li hodnota proměnné `MQMD.Expiry` MQEI_UNLIMITED, nastavte ji na maximální hodnotu pro pole AMQP ttl.
první-nabyvatel	ne	V produktu `mq_amqp.Hdr.Fac`, je-li nastaven, nebo nepravda jinak.
počet doručení	0	V poli `mq_amqp.Hdr.Dct`, je-li nastaveno, nebo jinak 0.

delivery-anotace

Podle potřeby nastavte kanál AMQP.

anotace-zprávy

Nezahrnuto.

vlastnosti

Pokud jsou tyto vlastnosti nastaveny, vlastnosti **properties** budou beze změn z ekvivalentních vlastností produktu `mq_amqp.Prp` . Pokud zpráva nebyla původně zprávou AMQP (to znamená, že typ PutApplnení MQAT_AMQP), vygeneruje se sekce vlastností podle popisu v následující tabulce:

Tabulka 81. Mapování polí vlastností

Název	Hodnota
id-zprávy	`MQMD.MsgId` je nastaven jako binární.
id-uživatele	Formát UTF-8 v `MQMD.UserIdentifier` je nastaven jako binární v pořadí bajtů sítě.
na	Fronta, ze které byla zpráva vydána, nebo pro publikaci řetězec tématu.
předmět	Není nastaveno.
odpověď na	`MQMD.ReplyToQ` , pokud není prázdné, jinak není nastaveno.

Tabulka 81. Mapování polí vlastností (pokračování)	
Název	**Hodnota**
ID korelace	`MQMD.CorrelId` je nastaven jako binární, není-li prázdný, jinak není nastaven.
Content-Type	Není nastaveno.
kódování obsahu	Není nastaveno.
absolutní-doba-vypršení	Není nastaveno.
creation-čas	Pole `MQMD.PutDate` a `MQMD.PutTime` se používají ke generování časového razítka.
id-skupiny	Není nastaveno.
skupina-posloupnost	Není nastaveno.
id-na-skupinu-odpovědi	Není nastaveno.

vlastnosti aplikace

Všechny vlastnosti IBM MQ ve skupině "usr" se přidávají jako **vlastnosti aplikace**.

tělo

Kanál AMQP provádí převod s převedováním, aby převedl informační obsah produktu IBM MQ do sady UTF-8.

Pokud informační obsah IBM MQ neobsahuje zprávu AMQP, pak je informační obsah IBM MQ v těle nastaven jako jednoduchá řetězcová datová sekce pro formát MQFMT_STRING (zadaná konverze na UTF-8 byla úspěšná), nebo jako jediná binární sekce dat.

Je-li zahrnuta zpráva o formátu AMQP, pak je tato zpráva nastavena jako tělo. Všechna záhlaví produktu IBM MQ (nezahrnující vlastnosti zpráv vrácené v popisovači zprávy) před zprávou AMQP jsou předřazena jako binární hodnota, pokud je tělo posloupnost AMQP. Jinak budou záhlaví IBM MQ zahozena.

zápatí

Není zahrnuto žádné zápatí.

Související informace

MQMD-deskriptor zprávy
Vytvoření a použití kanálů AMQP
Zabezpečení klientů AMQP

ULW V 9.0.0 Mapování polí AMQP na pole IBM MQ (příchozí zprávy)

Pokud kanál AMQP přijme zprávu a vloží ji do produktu IBM MQ, šíří některé atributy zprávy AMQP do ekvivalentních atributů zpráv produktu IBM MQ .

Při mapování příchozí zprávy AMQP platí následující omezení:

- Je-li pole `message-id` nebo `correlation-id` v části vlastností uuid nebo ulong, zpráva se odmítne.
- Všechny `message-annotations` způsobí, že se zpráva odmítne.
- Sekce `delivery-annotations` a `footer` jsou povoleny, ale nešíří se do zprávy IBM MQ .

Následující dílčí sekce zobrazují výraz IBM MQ zprávy AMQP.

deskriptor zprávy

Tabulka 82. Deskriptor zpráv pro zprávu AMQP	
Pole	**Hodnota**
StrucId	ID_STRUKTURY MQM_STRUCT
Verze	MQMD_VERSION_1
Sestava	MQRO_NONE
MsgType	MQM_DATAGRAM
Vypršení	Hodnota převzata z pole `ttl` v záhlaví zprávy AMQP
Zpětná vazba	MQFB_NONE
Kódování	MQENC_NORMAL
CodedCharSetId	1208 (UTF-8)
Formát	Viz Informační obsah
Priorita	Hodnota převzata z pole `priority` v záhlaví zprávy AMQP. Je-li nastavena, omezena na maximum 9. Není-li nastaven, použije se výchozí hodnota 4.
Trvání	Je-li pole `durable` v záhlaví zprávy AMQP nastaveno na hodnotu true, nastavte na hodnotu MQPER_PERSISTENT. Jinak nastavte na hodnotu MQPER_NOT_PERSISTENT.
MagId	Správce front alokuje jedinečný 24bajtový MsgId.
Korell1	Hodnota převzata z pole `correlation-id` ve vlastnostech AMQP, pokud je nastavena. Nastavte na 24bajtovou binární hodnotu. Jinak nastavte na hodnotu MQCI_NONE/.
BackoutCount	0
ReplyToQ	""
ReplyToQMgr	""
UserIdentifier	Nastaveno na identifikátor ověřeného uživatele, který je připojen k kanálu AMQP
AccountingToken	MQACT_NONE
ApplIdentityData	hexadecimální řetězec. Nastaveno na posledních 8 bajtů identifikátoru připojení MQ kanálu AMQP.
PutApplType	MQAT_AMQP
PutApplName	
PutDate	Hodnota převzata z pole `creation-time` vlastností AMQP, pokud je nastavena. Jinak nastaveno na aktuální datum.
PutTime	Hodnota převzata z pole `creation-time` vlastností AMQP, pokud je nastavena. Jinak nastaveno na aktuální čas.
ApplOriginData	""

Vlastnosti zprávy

Existují dva důvody pro nastavení vlastností zprávy:

- Chcete-li povolit, aby části zprávy AMQP procházeli prostřednictvím správce front, aniž byste ovlivnili informační obsah zprávy.
- Chcete-li povolit výběr produktu `application-properties`.

V následující tabulce jsou uvedeny vlastnosti, které jsou nastaveny ze zprávy AMQP:

Tabulka 83.

Název vlastnosti	**Název MQRFH2**	**Typ**	**Popis**
AMQPListener	mq_amqp.Lis	ŘETĚZEC MQTYPE_STRING	Identifikační řetězec pro kanál AMQP. Používá se k vygenerování zprávy, aby zúčastněné strany mohly zjistit, kterou verzi zpráva vložila (například tým služeb při diagnostikování problémů). Hodnota není ověřována správcem front a nesmí být dokumentována externě.
AMQPVersion	mq_amqp.Ver	ŘETĚZEC MQTYPE_STRING	Verze zprávy AMQP. Není-li přítomna, předpokládá se hodnota "1.0". Hodnota není ověřována správcem front.
AMQPClient	mq_amqp.Cli	ŘETĚZEC MQTYPE_STRING	Identifikační řetězec pro rozhraní API. Používá se k odeslání zprávy AMQP do kanálu, aby zúčastněné strany mohly říci, která verze tuto zprávu vložila (například tým služeb při diagnostikování problémů). Hodnota není ověřována správcem front a je třeba ji externě dokumentovat.
AMQPDurovatelný	mq_amqp.Hdr.Dur	LOGICKÁ HODNOTA MQTYPE_BOOLEAN	Hodnota pole `durable` v záhlaví zprávy AMQP, je-li nastavena.
Priorita AMQPPriority	mq_amqp.Hdr.Pri	MQTYPE_INT32	Hodnota pole `priority` v záhlaví zprávy AMQP, je-li nastavena.
AMQPTtl.	mq_amqp.Hdr.Ttl	MQTYPE_INT64	Hodnota pole `ttl` v záhlaví zprávy AMQP, je-li nastavena.
AMQPFirstAcquirer	mq_amqp.Hdr.Fac	LOGICKÁ HODNOTA MQTYPE_BOOLEAN	Hodnota pole `first-acquirer` v záhlaví zprávy AMQP, je-li nastavena.
AMQPDeliveryCount	mq_amqp.Hdr.Dct	MQTYPE_INT64	Hodnota pole `delivery-count` v záhlaví zprávy AMQP, je-li nastavena.
AMQPMsgId	mq_amqp.Prp.Mid	ŘETĚZEC MQTYPE_STRING	Hodnota pole `message-id` ve vlastnostech AMQP, je-li nastavena jako řetězec.
		ŘETĚZEC MQTYPE_BYTE_STRING	Hodnota pole `message-id` ve vlastnostech AMQP, je-li nastavena jako bajtový řetězec.
AMQPUserId	mq_amqp.Prp.Uid	ŘETĚZEC MQTYPE_BYTE_STRING	Hodnota pole `user-id` ve vlastnostech AMQP, je-li nastavena.

Tabulka 83. (pokračování)

Název vlastnosti	Název MQRFH2	Typ	Popis
AMQPDo	mq_amqp.Prp.To	ŘETĚZEC MQTYPE_STRING	Hodnota pole `to` ve vlastnostech AMQP, je-li nastavena.
AMQPSubject	mq_amqp.Prp.Sub	ŘETĚZEC MQTYPE_STRING	Hodnota pole `subject` ve vlastnostech AMQP, je-li nastavena.
AMQPReplyTo	mq_amqp.Prp.Rto	ŘETĚZEC MQTYPE_STRING	Hodnota pole `reply-to` ve vlastnostech AMQP, je-li nastavena.
AMQPCorrelationId	mq_amqp.Prp.Cid	ŘETĚZEC MQTYPE_STRING	Hodnota pole `correlation-id` ve vlastnostech AMQP, je-li nastavena jako řetězec.
		ŘETĚZEC MQTYPE_BYTE_STRING	Hodnota pole `correlation-id` ve vlastnostech AMQP, je-li nastavena jako bajtový řetězec.
AMQPContentType	mq_amqp.Prp.Cnt	ŘETĚZEC MQTYPE_STRING	Hodnota pole `content-type` ve vlastnostech AMQP, je-li nastavena.
AMQPContentEncoding	mq_amqp.Prp.Cne	ŘETĚZEC MQTYPE_STRING	Hodnota pole `content-encoding` ve vlastnostech AMQP, je-li nastavena.
AMQPAbsoluteExpiryČas	mq_amqp.Prp.Aet	ŘETĚZEC MQTYPE_STRING	Hodnota pole `absolute-expiry-time` ve vlastnostech AMQP, je-li nastavena.
AMQPCreationTime	mq_amqp.Prp.Crt	ŘETĚZEC MQTYPE_STRING	Hodnota pole `creation-time` ve vlastnostech AMQP, je-li nastavena.
AMQPGroupId	mq_amqp.Prp.Gid	ŘETĚZEC MQTYPE_STRING	Hodnota pole `group-id` ve vlastnostech AMQP, je-li nastavena.
AMQPGroupSequence	mq_amqp.Prp.Gsq	MQTYPE_INT64	Hodnota pole `group-sequence` ve vlastnostech AMQP, je-li nastavena.
AMQPReplyToGroupId	mq_amqp.Prp.Rtg	ŘETĚZEC MQTYPE_STRING	Hodnota pole `reply-to-group-id` ve vlastnostech AMQP, je-li nastavena.

Každá z vlastností aplikace ze zprávy AMQP je nastavena jako vlastnost zprávy produktu IBM MQ . Sekce `application-properties` musí být rozředěna identicky bajtová-pro-bajt, a proto platí následující omezení:

- Pokud je vlastnost aplikace odmítnuta kódem ověření MQSETMP, zpráva, která má být odmítnuta. Příklad:
  - Název vlastnosti je omezen délkou na MQ_MAX_PROPERTY_NAME_LENGTH.
  - Název vlastnosti musí odpovídat pravidlům definovaným ve specifikaci jazyka Java pro identifikátory Java.
  - Název vlastnosti nesmí začínat JMS nebo `usr.`JMS s výjimkou dokumentovaných vlastností JMS, které lze nastavit.
  - Název vlastnosti nesmí být klíčovým slovem SQL.
- Vlastnost aplikace obsahující znak Unicode U+002E (".") způsobí, že zpráva bude odmítnuta. Vlastnost musí být ve vlastnostech "usr" vlastností použitých platformou JMS výrazná.
- Podporovány jsou pouze hodnoty null, boolean, byte, short, int, long, float, double, binary a string. Vlastnost aplikace s jakýmkoli jiným typem způsobí odmítnutí zprávy.

informační obsah

- Pro prostředí AMQP bodys jednoduchou binární datovou částí jsou binární data (kromě bitů AMQP) vložena jako informační obsah produktu IBM MQ , s formátem MQFMT_NONE.
- Pro AMQP bodys jednou datovou částí řetězce je řetězcová data (kromě bitů AMQP) vložena jako informační obsah IBM MQ , s formátem MQFMT_STRING.
- Jinak příkaz AMQP body vytvoří informační obsah ve formátu MQFMT_AMQP s formátem MQFMT_AMQP.

Související informace

Vytvoření a použití kanálů AMQP
Zabezpečení klientů AMQP

ULW V 9.0.0 Spolehlivost doručení zpráv se AMQP

Existují čtyři funkce produktu IBM MQ Light API, které umožňují řídit spolehlivost doručování zpráv do aplikací MQ Light a AMQP a z nich.

Patří mezi ně:

- “Kvalita zpráv služby (QOS)” na stránce 664
- “Automatické potvrzení odběratele” na stránce 665
- “Doba platnosti odběru” na stránce 665
- “Trvalost zpráv” na stránce 665

Kvalita zpráv služby (QOS)

Produkt MQ Light API nabízí dvě úrovně kvality služby:

- Nejvíce jednou
- Nejméně jednou

Můžete si vybrat kvalitu služby, kterou mají vydavatelé a odběratelé používat.

Používáte-li klienta MQ Light , nastavte volbu klienta nebo odběru **qos** na hodnotu *QOS_AT_MOST_ONCE* nebo *QOS_AT_LEAST_ONCE*.

Používáte-li jiného klienta AMQP, nastavte atribut **`settled`** přenosového rámce (pro vydavatele) nebo rámec odebrání (pro odběratele) na hodnotu *true* nebo *false*, v závislosti na kvalitě služby, kterou chcete dosáhnout.

Kvalita služby určuje, kdy je zpráva vyřazena ze strany `sending` konverzace.

Publikování

Pokud vydavatel zvolí **QOS** *0* (nejvýše jednou), vydavatel nečeká na potvrzení od správce front, než bude vyřazena jeho kopie zprávy.

Pokud se připojení ke správci front nezdaří před dokončením odeslání, zpráva nemusí být přijata odběrateli.

Pokud vydavatel zvolí **QOS** *1* (alespoň jednou), vydavatel čeká, až správce front potvrdí, že zpráva byla zapsána do front odběratele před tím, než bude vyřazována její kopii zprávy.

Pokud dojde k selhání připojení ke správci front během odesílání, vydavatel zprávu znovu odešle, jakmile se znovu připojí ke správci front.

přihlášení odběru

Pokud odběratel zvolí **QOS** *0* , správce front nečeká na potvrzení od odběratele před tím, než bude vyřazením kopie zprávy čekat.

Pokud se připojení k odběrateli nezdaří, než odběratel obdrží zprávu, může dojít ke ztrátě této zprávy.

Pokud odběratel zvolí produkt **QOS** *1* , čeká správce front na potvrzení od odběratele před tím, než bude vyřazením kopie zprávy čekat.

Pokud se připojení k odběrateli nezdaří, než odběratel obdrží zprávu, zpráva je uchována správcem front. Správce front znovu odešle zprávu odběrateli, jakmile se správce front znovu připojí, nebo jinému odběrateli, pokud je odběr sdílen.

Automatické potvrzení odběratele

Pokud odběratel zvolí **QOS** *1* (alespoň jednou), musí potvrdit příjem každé zprávy předtím, než správce front zruší svou kopii. Odběratel se může rozhodnout, kdy potvrdí zprávy.

Při nastavení parametru **auto-confirm** na hodnotu *true*klient MQ Light automaticky potvrdí doručení každé zprávy, jakmile klient úspěšně obdrží zprávu po síti.

To zajišťuje, že pokud dojde k selhání sítě, zpráva se znovu doručí do aplikace. Nicméně je stále možné, aby aplikace přišla o zprávu, pokud aplikace selže mezi klientem produktu MQ Light , který potvrzuje zprávu, a aplikací, které ji zpracovávají.

Při nastavení parametru **auto-confirm** na hodnotu *false*klient produktu MQ Light automaticky nepotvrdí doručení zprávy, ale ponechá jej aplikaci, aby rozhodla, kdy by měla být potvrzena.

To umožňuje aplikaci provést aktualizaci na externí prostředek, jako je databáze nebo soubor, před potvrzením správce front, že zpráva byla nyní zpracována a může být vyřazena.

Doba platnosti odběru

Když se aplikace přihlašuje k odběru, zvolí si, zda odběr a místo určení, kde jsou zprávy uloženy pro daný odběr, budou existovat i po odpojení aplikace.

Volba odběru MQ Light **ttl** se používá k určení času (v milisekundách), po který odběr bude existovat po odpojení aplikace. Pokud se aplikace znovu připojí před touto dobou, bude obnoven odběr a aplikace může i nadále spotřebovávat zprávy z tohoto odběru.

Pokud doba životnosti uplyne bez opětovného navázání spojení s aplikací, bude odběr odebrán a všechny zprávy uložené v jejím cíli budou ztraceny, i když jsou to trvalé zprávy.

Pokud je důležité neztrácejte zprávy, musíte pro aplikaci určit dobu života, která bude dostatečně vysoká, aby se během výpadku neztratily zprávy.

Trvalost zpráv

Perzistence zpráv je řízena aplikacemi publikování a odběru a konfigurací objektů témat IBM MQ .

Pokud odběratel AMQP používá produkt **QOS** *0* (maximálně jednou) a vytvoří netrvalý odběr, kanál AMQP vždy vloží přechodné zprávy do fronty odběratele bez ohledu na další volby popsané v následujícím textu.

Všimněte si, že pokud je správce front zastaven jak odběr, tak i zprávy, které budou ztraceny.

Pokud vydavatel AMQP nastaví záhlaví AMQP **durable** na hodnotu *true*, kanál AMQP umístí trvalé zprávy do front odběratele.

Je-li správce front z nějakého důvodu zastaven, jsou při restartování správce front k dispozici pro odběratele zprávy.

Není-li záhlaví **durable** nastaveno, kanál AMQP zvolí perzistenci publikovaných zpráv na základě atributu **DEFPSIST** příslušného objektu tématu IBM MQ .

Ve výchozím nastavení je to SYSTEM.BASE.TOPIC, který používá atribut **DEFPSIST** s hodnotou *NO* (non-persistent).

Upozornění: Pozdější verze klienta MQ Light nepodporují nastavení trvalého záhlaví AMQP.

Související informace

Vytvoření a použití kanálů AMQP

Zabezpečení klientů AMQP

ULW Topologie pro klienty AMQP s produktem IBM MQ

Ukázkové topologie vám pomohou při vývoji klientů AMQP, aby mohli pracovat s produktem IBM MQ.

Související informace

Vytvoření a použití kanálů AMQP

Zabezpečení klientů AMQP

ULW Klienti AMQP komunikují prostřednictvím produktu IBM MQ

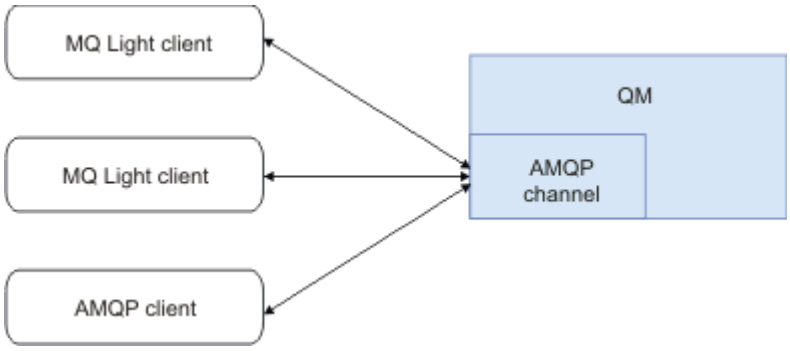
Můžete použít produkt IBM MQ jako poskytovatele systému zpráv pro produkt IBM MQ Light nebo libovolnou aplikaci, která splňuje podmínky AMQP 1.0. Ačkoli se klient AMQP 1.0 může připojit ke kanálu AMQP, některé funkce AMQP nejsou podporovány, například transakce nebo více relací.

Definováním jednoho nebo více kanálů AMQP se klienti AMQP 1.0 mohou připojit ke správci front a odesílat zprávy na řetězec tématu. Klienti se mohou také přihlásit k odběru zpráv, které odpovídají vzoru.

V následujícím scénáři jsou jedinými aplikacemi odesílající a přijímající zprávy aplikace MQ Light nebo AMQP 1.0 .

Aplikace mohou zvolit, zda jsou místa určení vytvořená přihlášením k odběru řetězce tématu trvalá, takže se zprávy neztratí, pokud aplikace dočasně ztratí své připojení ke správci front.

Aplikace si také mohou vybrat, jak dlouho se budou uchovávat zprávy před tím, než budou vymazány z cíle.



Související informace

Vytvoření a použití kanálů AMQP

Zabezpečení klientů AMQP

ULW Klienti AMQP vyměňují zprávy s aplikacemi produktu IBM MQ

Definováním a spuštěním kanálu AMQP mohou aplikace produktu MQ Light nebo AMQP 1.0 publikovat zprávy přijaté existujícími aplikacemi produktu MQ . Zprávy publikované prostřednictvím kanálu AMQP se všechny odesílají do témat MQ , nikoli do front MQ . Aplikace MQ , která vytvořila odběr pomocí volání rozhraní MQSUB API, přijímá zprávy publikované aplikacemi AMQP 1.0 za předpokladu, že řetězec tématu nebo objekt tématu používaný aplikací produktu MQ odpovídá řetězci tématu publikovaném klientem AMQP.

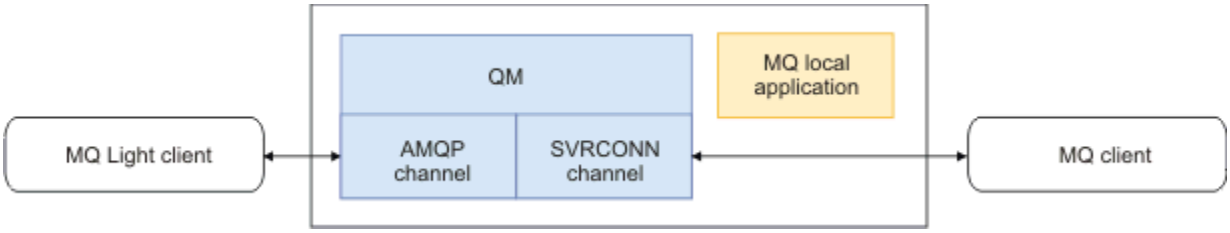
Data zprávy AMQP, atributy a vlastnosti jsou nastaveny na zprávě MQ přijaté aplikací MQ . Další informace o mapování zprávy AMQP na MQ naleznete v tématu Mapování polí AMQP na pole produktu IBM MQ (příchozí zprávy).

Pokud aplikace MQ vytvořila trvalý odběr, zprávy publikované aplikací AMQP jsou uloženy ve frontě, která podporuje odběr. Zprávy jsou poté přijaty aplikací MQ , jakmile aplikace obnoví svůj odběr. Pokud aplikace AMQP uvádí dobu životnosti zprávy a aplikace MQ se znovu nepřipojí v době, kdy má být aktivní, vyprší platnost zprávy z fronty.

Aplikace MQ Light nebo AMQP 1.0 mohou rovněž přijímat zprávy, které jsou publikovány existujícími aplikacemi MQ . Zprávy publikované aplikacemi produktu MQ do tématu MQ nebo řetězce tématu jsou přijímány aplikací AMQP 1.0 za předpokladu, že aplikace byla přihlášena k odběru vzorku tématu, který odpovídá publikovanému řetězci tématu.

Pokud aplikace AMQP 1.0 určuje časovou hodnotu odběru a aplikace AMQP se odpojí po dobu životnosti, vypršela platnost odběru ze správce front a všechny zprávy uložené ve frontě odběru se ztratí.

Pole MQMD, vlastnosti zprávy a data aplikací jsou nastavena na zprávu AMQP přijatou aplikací AMQP. Další informace o mapování zpráv produktu MQ na zprávu AMQP naleznete v tématu Mapování polí AMQP na pole produktu IBM MQ (odchozí zprávy).



Související informace

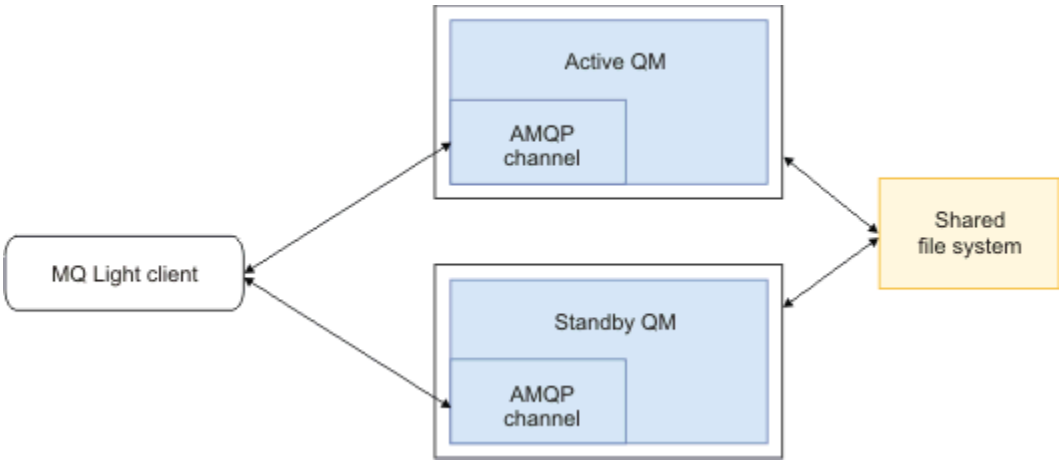
Vytvoření a použití kanálů AMQP

Zabezpečení klientů AMQP

ULW Konfigurace klienta AMQP pro vysokou dostupnost

Můžete nakonfigurovat aplikace MQ Light nebo AMQP 1.0 pro připojení k aktivní instanci správce front IBM MQ s více instancemi a překonání selhání na instanci správce front s více instancemi ve správci front s vysokou dostupností (HA). Chcete-li tuto akci provést, nakonfigurujte aplikaci AMQP se dvěma adresami IP a s dvojicemi portů.

Rozhraní API produktu MQ Light můžete nakonfigurovat pomocí vlastní funkce, která se volá, pokud klient ztratí své připojení k serveru. Funkce se může připojit k alternativní adrese IP, například v rezervním správci front MQ nebo na původní adresu IP. Pokud klient podporuje konfiguraci více koncových bodů připojení, nakonfigurujte pro další klienty AMQP konfiguraci aplikace se dvěma dvojicemi hostitel-port a použijte funkce opětného připojení poskytnuté knihovnou AMQP k přepnutí do rezervního správce front.



Související informace

Vytvoření a použití kanálů AMQP

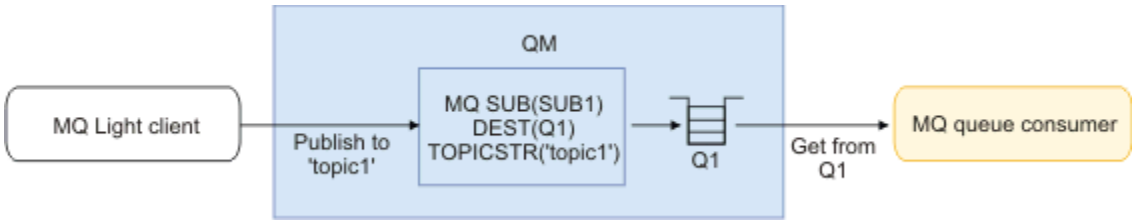
Zabezpečení klientů AMQP

ULW Konfigurace publikování/odběru pro klienty AMQP

Klienti AMQP mohou publikovat do tématu s odběrem produktu IBM MQ , který směruje zprávy do fronty IBM MQ načtené existující aplikací. Chcete-li, aby aplikace MQ Light nebo AMQP 1.0 odeslaly zprávy do

existující aplikace produktu IBM MQ , která je konfigurována ke čtení z fronty, musíte ve správci front definovat spravovaný odběr produktu IBM MQ .

Nakonfigurujte odběr a použijte vzorek tématu, který odpovídá řetězci tématu používannému aplikací AMQP. Nastavte cíl odběru na název fronty, ze které aplikace IBM MQ získává nebo prochází zprávy.



Související informace

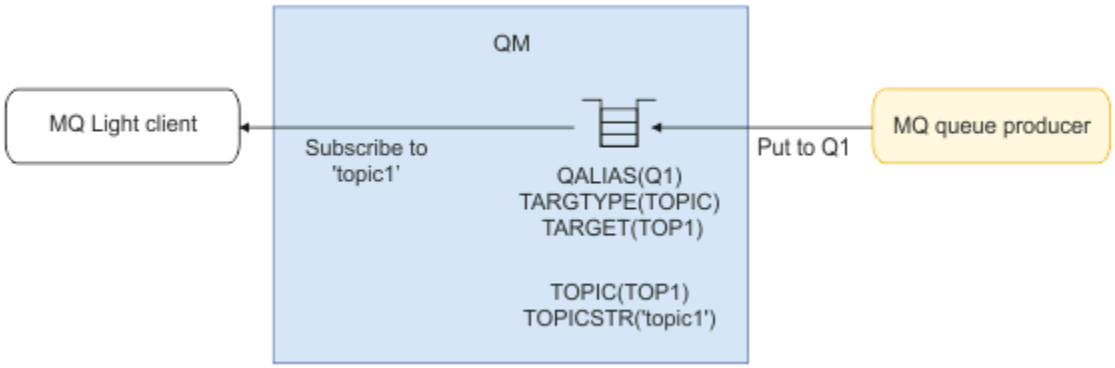
Vytvoření a použití kanálů AMQP

Zabezpečení klientů AMQP

ULW Klient AMQP s použitím aliasu fronty pro příjem zpráv z aplikace produktu IBM MQ

Klient AMQP se může přihlásit k odběru tématu a přijímat zprávy zařazené do fronty aliasů prostřednictvím aplikace produktu IBM MQ . Chcete-li, aby aplikace MQ Light nebo AMQP 1.0 přijímala zprávy z existující aplikace produktu IBM MQ , která je konfigurována pro vkládání zpráv do fronty, je třeba definovat alias fronty (QALIAS) ve správci front.

Alias fronty musí mít stejný název jako fronta, kterou se aplikace IBM MQ otevírá pro vložení. Alias fronty musí určovat základní typ TOPIC a základní objekt objektu tématu produktu IBM MQ , který má řetězec tématu, který odpovídá vzoru tématu přihlášeným k odběru aplikace AMQP.



Související informace

Vytvoření a použití kanálů AMQP

Zabezpečení klientů AMQP

ULW Klient AMQP odesílá požadavky na aplikační server a spotřebovává jejich odezvy.

Klient MQ Light nebo jiný klient AMQP může odesílat požadavky na objekt typu message-driven bean spuštěný na aplikačním serveru a přijímat odpovědi z tématu odpovědi. Produkt IBM MQ podporuje nastavení zpráv AMQP 1.0 ve zprávách, které publikuje produkt IBM MQ , na téma. Je-li zpráva AMQP publikována s atributem odpovědi na sadu atributů, hodnota pole odpovědi na pole je nastavena jako vlastnost JMS pro spotřebitele služby JMS, kteří mají být obdrženi. Toto nastavení umožňuje spotřebitelům služby JMS číst odpovědi na téma odpovědi ze zprávy a odeslat zprávu odpovědi zpět klientovi AMQP.

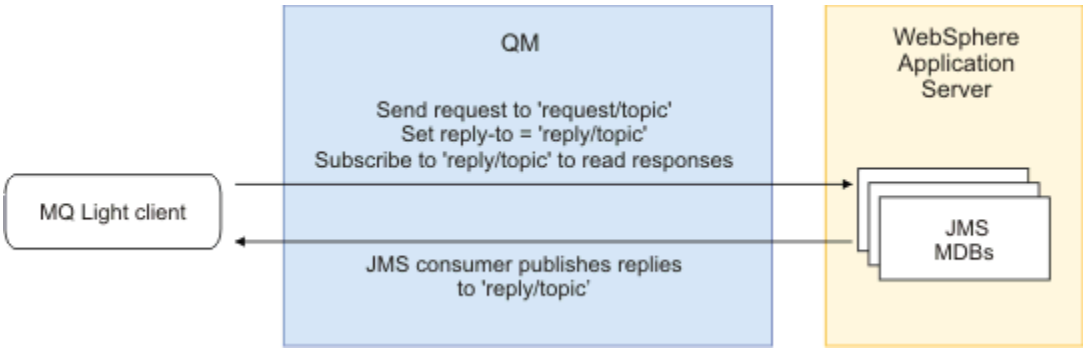
Vlastnost JMS je **JMSReplyTo**. Odpověď AMQP na řetězec musí být jeden z následujících typů:

- řetězec tématu. Například: `'reply/topic'`

- Adresa URL adresy AMQP ve formuláři `amqp://host:port/[topic-string]`. Například `amqp://localhost:5672/reply/topic`

If you specify an AMQP address URL as the reply-to field, everything except the topic-string at the end of the URL is removed before setting the **JMSReplyTo** property.

Další informace o mapování z odpovědi AMQP-na adresu do vlastnosti **JMSReplyTo** naleznete v tématu Mapování polí AMQP na pole produktu IBM MQ (příchozí zprávy)



Související informace

Vytvoření a použití kanálů AMQP

Zabezpečení klientů AMQP

ULW Implementace aplikací produktu MQ Light do prostředí produktu IBM MQ s lokální verzí

IBM MQ podporuje rozhraní API systému zpráv produktu IBM MQ Light , takže můžete použít produkt IBM MQ k implementaci aplikace produktu MQ Light do prostředí produktu IBM MQ .

Aplikace produktu MQ Light můžete implementovat do správce front produktu IBM MQ , a umožnit tak aplikacím produktu MQ Light komunikovat s existujícími podnikovými aplikacemi, které jsou již připojeny k produktu IBM MQ, jak je znázorněno v následujícím diagramu:

Aplikace produktu MQ Light mohou sdílet prostor tématu s existujícími aplikacemi produktu IBM MQ , což jim umožňuje interakci s existujícími podnikovými systémy.

Související informace

Vytvoření a použití kanálů AMQP

Zabezpečení klientů AMQP

Vývoj aplikací REST s produktem IBM MQ

Můžete vyvíjet aplikace REST pro odesílání a příjem zpráv. Produkt IBM MQ podporuje různá rozhraní REST API v závislosti na platformě a schopnosti.

Podporovaná volba produktu IBM MQ si můžete vybrat, zda chcete odesílat zprávy a přijímat zprávy od produktu IBM MQ pomocí služby REST, a to pomocí následujících voleb:

- IBM MQ messaging REST API
- IBM MQ most pro protokol HTTP
- IBM z/OS Connect EE
- IBM Integration Bus
- DataPower

V 9.0.4 IBM MQ messaging REST API

messaging REST API se standardně dodává s IBM MQ z IBM MQ 9.0.4 a je ve výchozím nastavení povolený. Produkt messaging REST API můžete použít k odesílání a přijímání zpráv produktu IBM MQ ve formátu prostého textu.

Aplikace mohou vydat zprávu HTTP POST pro odeslání zprávy do produktu IBM MQnebo HTTP DELETE za účelem destruktivního získání zprávy od produktu IBM MQ. Podpora je poskytována pro řadu různých záhlaví HTTP, které lze použít k nastavení obecných vlastností zpráv.

Produkt messaging REST API je plně integrován se zabezpečením IBM MQ . Uživatelé musí být ověřeni na webovém serveru mqweb a musí být členem role MQWebUser .

Další informace naleznete v tématu Systém zpráv s použitím produktu REST API.

IBM MQ bridge for HTTP

Most produktu IBM MQ pro protokol HTTP je aplikace JEE, kterou lze instalovat do vhodného běhového prostředí a poskytuje základní REST API na obou frontách a v tématech hostovaných jedním správcem front MQ .

Aplikace mohou zadáním požadavku HTTP POST do mostu odeslat zprávu do produktu IBM MQ, protokol HTTP GET pro prohlížení zprávy z produktu IBM MQnebo protokol HTTP DELETE k destruktivnímu získání zprávy z produktu IBM MQ. Podpora je poskytována pro řadu různých záhlaví HTTP, které lze použít k nastavení vlastností zpráv.

Poznámka: V produktu IBM MQ 8.0je funkce IBM MQ bridge for HTTP zamítnuta. Volba IBM messaging REST API poskytnutá z produktu IBM MQ 9.0.4 by měla být použita jako alternativa.

Další informace naleznete v tématu Vývoj webových služeb pomocí mostu produktu IBM MQ pro protokol HTTP.

IBM z/OS Connect EE

IBM z/OS Connect EE (zCEE) je produkt z/OS , který vám umožňuje sestavit rozhraní REST API na stávajících aktivech systému z/OS , jako jsou transakce systému CICS nebo IMS , a fronty a témata produktu IBM MQ . Existující aktivum systému z/OS je skryto před uživatelem. To vám umožňuje REST povolit vaše aktiva, aniž byste je měnili, nebo jakoukoli z existujících aplikací, které je používají.

S pomocí zCEE můžete snadno sestavit REST API , který bude odesílat a přijímat data JSON a volitelně transformovat tato data do více tradičních jazykových struktur očekávaných mnoha aplikacemi sálových počítačů. Například zakladače COBOL.

Editor rozhraní Eclipse založený na rozhraní API lze použít k sestavení komplexního rozhraní RESTful API pro použití parametrů dotazu a segmentů cesty adresy URL a k manipulaci s formátem JSON, jak to prochází běhovým prostředím zCEE .

zCEE lze použít k vystavení front IBM MQ a témat jako služby. Jsou podporovány dva různé typy služeb:

- Jednosměrné služby: poskytovaná funkce je podobná té, která je poskytována s použitím mostu IBM MQ pro protokol HTTP, kde HTTP POST odešle zprávu, destruktivně získá zprávu HTTP DELETE. Hlavní výhody z mostu jsou zabudovány v podpoře konverze dat a že můžete použít editor rozhraní API k sestavení komplexnějšího rozhraní RESTful API.
- Two way services: this provides a REST API on top of a pair of queues used by a backend request-response style application. Callers vydá HTTP POST do dvoucestné služby a odešle data ve formátu JSON. Tato data jsou vložena do fronty požadavků, kde je zpracována back-endové aplikací a odezvou umístěná na frontu odpovědí. Tato odpověď je načtena a odeslána zpět volajícímu jako tělo odezvy testu POST.

Produkt zCEE je podporován v systémech IBM MQ 8 a novějších. Další informace naleznete v tématu IBM MQ for z/OS Service Provider for z/OS Connect.

IBM Integration Bus

Produkt IBM Integration Bus je špičková technologie integrace společnosti IBM, kterou lze použít k připojení aplikací a systémů dohromady bez ohledu na formáty zpráv a protokoly, které podporují.

Produkt IBM Integration Bus vždy podporuje IBM MQ a poskytuje uzly *HTTPInput* a *HTTPRequest* , které lze použít k vytvoření rozhraní RESTful na vrcholu IBM MQa mnoha dalších systémů, jako jsou databáze.

Produkt IBM Integration Bus lze použít k mnohem více než poskytnutí jednoduchého rozhraní REST v horní části produktu IBM MQ. Jeho schopnosti lze použít k poskytnutí rozšířené manipulace s informačním obsahem, obohacení informačního obsahu a mnoha dalšími vylepšeními jako součásti produktu REST API.

Další informace naleznete v ukázce technologie , která odkrývá JSON přes rozhraní REST nad aplikací produktu IBM MQ , která očekává informační obsah XML.

DataPower

Brána DataPower je samostatná vícekanálová brána, která pomáhá poskytovat zabezpečení, řízení, integraci a optimalizovaný přístup k řadě systémů, včetně IBM MQ. Je dodáván jak v hardwarových, tak i virtuálních faktorech.

Jedna z služeb, které poskytuje produkt DataPower , je víceprotokolová brána, která může přijímat vstup v jednom protokolu a generovat výstup v jiném protokolu. Produkt DataPower může být nakonfigurován tak, aby přijímal data HTTP (S) a přesměroval jej na IBM MQ přes připojení klienta, které lze použít k sestavení rozhraní REST na vrcholu IBM MQ. Další služby produktu DataPower , jako např. transformace, lze také použít ke zlepšení rozhraní REST.

Další informace naleznete v tématu Služba komunikační brány s více protokoly.

V 9.0.4 Zasílání zpráv pomocí produktu REST API

K odesílání a přijímání zpráv produktu IBM MQ můžete použít produkt messaging REST API . Informace jsou momentálně odesílány a přijímány z produktu messaging REST API ve formátu prostého textu.

Než začnete

Poznámka:

Ve výchozím nastavení je volba messaging REST API povolena. Chcete-li zabránit veškerům systému zpráv, můžete produkt messaging REST API zakázat ručně. Další informace o povolení nebo zakázání funkce messaging REST APInaleznete v části Konfigurace produktu messaging REST API.

Produkt messaging REST API je integrován se zabezpečením produktu IBM MQ . Volající musí být ověřen na webovém serveru mqweb a musí být členem role MQWebUser . Volající musí být také autorizován pro přístup k uvedené frontě. Další informace o zabezpečení pro REST API viz Zabezpečení konzoly IBM MQ Console a REST API.

Procedura

- “Začínáme s produktem messaging REST API” na stránce 672
- “Použití produktu messaging REST API” na stránce 674
- “Omezení systému zpráv REST API” na stránce 675
- REST API ošetření chyb
- REST API zjišťování
- REST API národní jazyková podpora

Související informace

Odkaz systému zpráv REST API

V 9.0.4 Začínáme s produktem messaging REST API

Než budete moci spustit messaging REST API , musíte nainstalovat správné komponenty, povolit REST API, nakonfigurovat zabezpečení a spustit mqweb server.

Než začnete

IBM i Na IBM iby měly být příkazy spuštěny v QSHELL.

Informace o této úloze

Procedura pro tuto úlohu se zaměřuje na rychlé zahájení práce s produktem messaging REST API. Kroky pro konfiguraci zásady zabezpečení popisují, jak nastavit základní registr uživatelů, ale existují další volby pro konfiguraci uživatelů a rolí. Další informace o konfiguraci zabezpečení pro produkt messaging REST APInaleznete v příručce IBM MQ Console and REST API security.

Poznámka: Chcete-li přistupovat k souboru `mqwebuser.xml` , musíte být privilegovaný uživatel .

Postup

1. Nainstalujte komponentu IBM MQ Console a REST API :
   - AIX V systému AIXnainstalujte sadu souborů `mqm.web.rte` .
   - Linux V systému Linuxnainstalujte komponentu produktu `MQSeriesWeb` . Další informace o instalaci komponent a funkcí v systému Linuxnaleznete v tématu Úlohy instalace produktu Linux.
   - Windows V systému Windowsnainstalujte funkci `Web Administration` . Další informace o instalaci komponent a funkcí v systému Windowsnaleznete v tématu Úlohy instalace produktu Windows.
   - z/OS V systému z/OSnainstalujte funkci `IBM MQ for z/OS Unix System Services Components` . Další informace o instalaci komponent a funkcí v systému z/OSnaleznete v tématu Úlohy instalace produktu z/OS.
2. Nakonfigurujte uživatele a role před tím, než budete moci použít produkt messaging REST API:
   a) Zkopírujte soubor `basic_registry.xml` z adresáře *MQ_INSTALLATION_PATH* `/web/mq/samp/configuration` .
   b) Ukázkový soubor XML umístěte do odpovídajícího adresáře:
      - ULW V systému UNIX, Linux, and Windows: *MQ_DATA_DIRECTORY*`/web/installations/` *installationName*`/servers/mqweb`
      - z/OS V systému z/OS: *WLP_user_directory*`/servers/mqweb`

      kde *WLP_user_directory* je adresář, který byl zadán při spuštění skriptu **crtmqweb.sh** za účelem vytvoření definice mqweb serveru.
   c) Přejmenujte ukázkový soubor XML na `mqwebuser.xml`.

      Poznámka: Tento přejmenovaný soubor nahrazuje existující soubor, který se také používá pro IBM MQ Console. Proto, pokud jste změnili soubor `mqwebuser.xml` pro IBM MQ Console, zkopírujte své změny do nového souboru XML, dříve než jej přejmenujete.
   d) Volitelné: Upravte soubor `mqwebuser.xml` a přidejte uživatele a skupiny. Přiřaďte tyto uživatele a skupiny k roli produktu `MQWebUser` , abyste byli autorizováni používat produkt messaging REST API. Role `MQWebAdmin` a `MQWebAdminRO` nejsou použitelné pro messaging REST API. Můžete také změnit hesla pro uživatele, kteří jsou nadefinovali při výchozím nastavení, a zakódovat nová hesla. Ujistěte se, že uživatelé mají oprávnění pro přístup k uvedené frontě. Další informace viz téma Konfigurace uživatelů a rolí.
3. Pomocí příkazu **setmqweb** povolte vzdálená připojení k mqweb serveru:

```
setmqweb properties -k httpHost -v hostname
```

kde parametr *název_hostitele* určuje adresu IP, název hostitele DNS (Domain Name Server) s příponou názvu domény nebo název hostitele DNS pro server, na kterém je nainstalován produkt IBM MQ . Chcete-li zadat všechna dostupná síťová rozhraní, použijte hvězdičku (*).

Upozornění: V 9.0.4 z/OS

Před zadáním příkazu **setmqweb** nebo **dspmqweb** v systému z/OS musíte nastavit proměnnou prostředí WLP_USER_DIR tak, aby proměnná ukazovala na konfiguraci serveru mqweb.

Chcete-li to provést, zadejte následující příkaz:

```
export WLP_USER_DIR=WLP_user_directory
```

kde *WLP_user_directory* je název adresáře, který je předán do `crtmqweb.sh`. Příklad:

```
export WLP_USER_DIR=/var/mqm/web/installation1
```

Další informace viz Vytvoření definice serveru Liberty.

4. Spusťte mqweb server, který podporuje REST API:
   - ULW Jako privilegovaný uživatelzadejte na příkazový řádek následující příkaz:

     ```
     strmqweb
     ```
   - z/OS V systému z/OSspusťte proceduru, kterou jste vytvořili v Úloha 29: Vytvořte proceduru pro server IBM WLP.

Jak pokračovat dále

1. Zvolte, jak se uživatelé produktu messaging REST API ověřují s pomocí mqweb serveru. Stejnou metodu pro všechny uživatele nemusíte používat. K dispozici jsou následující volby:
   - Nechte uživatele provést ověření pomocí základního ověření HTTP. V tomto případě je jméno uživatele a heslo zakódováno, ale není šifrováno a odesláno s každým požadavkem REST API na ověření a autorizaci uživatele pro tento požadavek. Má-li být toto ověření zabezpečené, je třeba použít zabezpečené připojení. To znamená, že musíte použít protokol HTTPS. Další informace viz Použití základního ověření HTTP pomocí rozhraní REST API.
   - Nechat uživatele ověřit pomocí ověření tokenu. V tomto případě uživatel zadá ID uživatele a heslo do prostředku produktu REST API `login` pomocí metody HTTP POST. Je vygenerován token LTPA, který umožňuje uživateli zůstat přihlášen a autorizován pro určitý časový interval. Má-li být toto ověření zabezpečené, je třeba použít zabezpečené připojení. To znamená, že musíte použít protokol HTTPS. Další informace naleznete v tématu Použití ověření pomocí tokenů se serverem REST API.
   - Nechte uživatele ověřovat certifikáty pomocí klientských certifikátů. V tomto případě uživatel nepoužije ID uživatele nebo heslo pro přihlášení k serveru messaging REST API, ale použije místo toho certifikát klienta. Další informace naleznete v tématu Použití ověření klientského certifikátu pomocí produktu REST API.
2. Nakonfigurujte nastavení produktu REST API , včetně povolení připojení HTTP, a změnu čísla portu. Další informace naleznete v tématu Konfigurace konzoly IBM MQ a produktu REST API.
3. Volitelně můžete nakonfigurovat sdílení prostředků různého původu pro REST API. Standardně nemůžete přistoupit k produktu REST API z webových prostředků, které nejsou hostovány na stejné doméně jako REST API. To znamená, že požadavky typu cross-origin nejsou povoleny. Sdílení CORS (Cross Origin Resource Sharing) můžete nakonfigurovat tak, aby bylo možné povolit požadavky na křížový původ ze zadaných adres URL. Další informace naleznete v tématu Konfigurace CORS pro produkt REST API.
4. Použijte REST API. Další informace viz “Použití produktu messaging REST API” na stránce 674a Odkaz na systém zpráv REST API.

Poznámka: Server mqweb můžete kdykoli zastavit pomocí příkazu **endmqweb** . Pokud však není spuštěn server mqweb, nelze použít REST API ani IBM MQ Console.

V 9.0.4 Použití produktu messaging REST API

Když používáte produkt messaging REST API, vyvoláte na adresách URL metody HTTP k odesílání a přijímání zpráv produktu IBM MQ . Metoda HTTP, například POST, představuje typ akce, která má být provedena na objektu, který je reprezentován adresou URL. Další informace o akci mohou být kódovány v parametrech dotazu. Informace o výsledku provedení akce mohou být vráceny jako tělo odpovědi HTTP.

Než začnete

Před použitím produktu messaging REST APIvezměte v úvahu tyto skutečnosti:

- Chcete-li používat produkt messaging REST API, musíte se ověřit na serveru mqweb. Ověření můžete provést pomocí základního ověření HTTP, ověření certifikátu klienta nebo ověření založeného na tokenech. Další informace o použití těchto metod ověření naleznete v příručce IBM MQ Console and REST API security.
- Produkt REST API rozlišuje velká a malá písmena. Například test HTTP POST na následující adrese URL má za následek chybu, pokud se správce front jmenuje `qmgr1`.

  ```
  /ibmmq/rest/v1/messaging/qmgr/QMGR1/queue/Q1/message
  ```

- Ne všechny znaky, které lze použít v názvech objektů produktu IBM MQ , lze v adrese URL přímo zakódovat. Chcete-li tyto znaky správně kódovat, je třeba použít příslušné kódování adresy URL:
  - Dopředné lomítko,/, musí být zakódováno jako %2F.
  - Znaménko procent,%, musí být zakódováno jako %25.

Informace o této úloze

Pokud k provedení akce systému zpráv na objektu fronty produktu IBM MQ používáte produkt REST API , je třeba nejprve sestavit adresu URL, která tento objekt reprezentuje. Každá adresa URL začíná předponou, která popisuje název hostitele a port, na který má být požadavek odeslán. Zbytek adresy URL popisuje konkrétní objekt nebo přenosovou cestu k danému objektu, označovaný jako prostředek.

Akce systému zpráv, která má být provedena na prostředku, definuje, zda adresa URL potřebuje parametry dotazu, či nikoli. Definuje také použitou metodu HTTP a informace o tom, zda jsou do adresy URL odeslány další informace nebo které z ní byly vráceny. Další informace mohou být součástí požadavku HTTP, nebo mohou být vráceny jako součást odezvy HTTP.

Po vytvoření adresy URL můžete odeslat požadavek HTTP do produktu IBM MQ. Požadavek můžete odeslat s použitím implementace HTTP, která je vestavěna do programovacího jazyka dle vašeho výběru. Požadavek můžete také odeslat pomocí nástrojů příkazového řádku, jako např. cURL, webového prohlížeče nebo webového prohlížeče add-on.

Důležité: Musíte provést kroky “1.a” na stránce 674 a “1.b” na stránce 675jako minimum.

Postup

1. Sestavit adresu URL:
   a) Začněte s touto předponou URL:

   ```
   https://host:port/ibmmq/rest/v1/messaging
   ```

   hostitel
   Určuje název hostitele nebo adresu IP, na které je messaging REST API k dispozici.

   Výchozí hodnota je `localhost`.

   Port
   Uvádí číslo portu HTTPS, které používá messaging REST API .

Výchozí hodnota je 9443.

Povolíte-li připojení HTTP, můžete místo HTTPS použít protokol HTTP. Další informace o povolení HTTP najdete v tématu Konfigurace portů HTTP a HTTPS.

Další informace o tom, jak určit adresu URL pro předponu, najdete v tématu Určení adresy URL prostoru REST API.

b) Přidejte frontu a přidružené prostředky správce front, které mají být použity pro systém zpráv, do cesty adresy URL.

V odkazu na systém zpráv mohou být segmenty proměnných v adrese URL identifikovány složenými závorkami, které ji obklopují *{ }*. Pro další informace viz /messaging/qmgr/{qmgrName}/queue/{queueName}/message.

Chcete-li například pracovat s frontou *Q1* přidruženou ke správci front *QM1*, přidejte do adresy URL předpony `/qmgr` a `/queue` , a vytvořte tak následující adresu URL:

```
https://localhost:9443/ibmmq/rest/v1/messaging/qmgr/QM1/queue/Q1/message
```

c) Volitelné: Přidejte do adresy URL volitelný parametr dotazu.

Přidat otazník,?, parametr dotazu, znak rovnítka =, a hodnota pro adresu URL.

Chcete-li například čekat maximálně 30 sekund na to, aby se další zpráva stala dostupnou, vytvořte následující adresu URL:

```
https://localhost:9443/ibmmq/rest/v1/messaging/qmgr/QM1/queue/Q1/message?wait=30000
```

d) Volitelné: Přidejte další volitelné parametry dotazu do adresy URL.

Přidejte ampersand, &, na adresu URL a pak zopakujte krok 1c.

2. Na adrese URL vyvolejte příslušnou metodu HTTP. Uveďte libovolný volitelný informační obsah zprávy a poskytněte odpovídající pověření zabezpečení pro ověření. Příklad:
   - Použijte implementaci HTTP/REST vašeho zvoleného programovacího jazyka.
   - Použijte nástroj, jako je doplněk prohlížeče klienta REST nebo cURL.

V 9.0.4 Omezení systému zpráv REST API

Před použitím produktu messaging REST APIvezměte v úvahu následující omezení:

- Rozhraní API v současné době nepodporuje systém zpráv publikování/odběru. Produkt messaging REST API podporuje pouze synchronní dvoubodový systém zpráv.
- Rozhraní API momentálně nepodporuje procházení z front. Zprávy jsou destruktivně přijaty ze fronty IBM MQ pomocí metody HTTP DELETE. Další informace viz DELETE.
- Při odesílání zprávy lze použít pouze textový obsah `UTF-8` . Zprávy jsou naformátovány jako formátované zprávy produktu IBM MQ verze `MQSTR` . Další informace najdete v tématu POST.

  Při příjmu zprávy jsou podporovány pouze formátované zprávy IBM MQ `MQSTR` . Následně jsou všechny zprávy přijaty pod synchronizačním bodem a všechny neošetřené zprávy jsou ponechány ve frontě.

  Frontu IBM MQ lze nakonfigurovat tak, aby přesunula tyto nezpracovatelné zprávy do alternativního cíle. Další informace naleznete v tématu Zpracování nezpracovatelných zpráv v produktu IBM MQ classes for JMS.
- Pokud používáte produkt Advanced Message Security (AMS) s produktem messaging REST API, všimněte si, že všechny zprávy jsou šifrovány pomocí kontextu mqweb serveru, nikoli kontextu uživatele, který tuto zprávu odesílá.
- Nové řádky v příchozích řetězcích jsou odebrány z operace HTTP POST. Aplikace REST

  neměly by používat nové řádky ve zprávách, které jsou odeslány nebo publikovány pomocí rozhraní API služby REST, protože budou ztraceny.

Vývoj webových služeb pomocí produktu IBM MQ bridge for HTTP

V produktu IBM MQ bridge for HTTPmohou klientské aplikace vyměňovat zprávy s produktem IBM MQ bez nutnosti instalovat produkt IBM MQ MQI client. Produkt IBM MQ můžete volat z libovolné platformy nebo jazyka s možností HTTP.

Informace o této úloze

Poznámka: V produktu IBM MQ 8.0je funkce IBM MQ bridge for HTTP zamítnuta. Volba IBM messaging REST API poskytnutá z produktu IBM MQ 9.0.4 by měla být použita jako alternativa.

Úvod do produktu IBM MQ bridge for HTTP

IBM MQ bridge for HTTP je webová aplikace Java, Enterprise Environment (JEE). Klienti HTTP mohou odesílat, procházet a odstraňovat zprávy z front produktu IBM MQ , aby mohli odesílat, procházet a odstraňovat zprávy z fronty produktu **POST**, **GET**a **DELETE** . Produkt IBM MQ bridge for HTTP není vhodný pro použití se zprávami, pokud je požadováno zajištěné doručení.

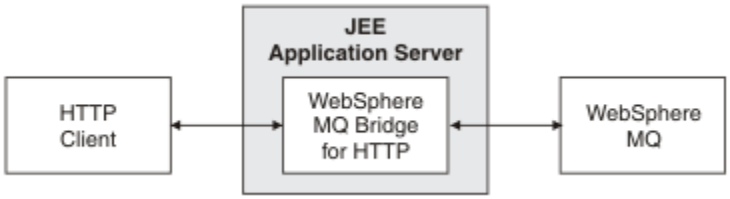
Výhody

S produktem IBM MQ bridge for HTTP můžete odesílat a přijímat zprávy produktu IBM MQ pomocí protokolu HTTP ze širokého spektra prostředí:

- Prostředí, která podporují protokol HTTP, ale ne produkt IBM MQ.
- Prostředí, která nemají dostatek úložného prostoru k instalaci produktu IBM MQ MQI client.
- Prostředí, která jsou příliš početná pro instalaci produktu IBM MQ MQI client na každý systém, který vyžaduje přístup k produktu IBM MQ.
- Webové aplikace, ze kterých chcete odesílat nebo přijímat zprávy bez kódování vlastního mostu do produktu IBM MQ.
- Webové aplikace, které chcete vylepšit, pomocí asynchronních metod, jako je technologie AJAX. Produkt IBM MQ bridge for HTTP zpřístupňuje fronty a témata produktu IBM MQ pomocí služby Representation State Transfer (REST) přes protokol HTTP.

Podporu HTTP lze používat s dvoubodovým systémem zpráv i s topologií systému zpráv typu publikování-odběr.

Jak podpora HTTP funguje?



Obrázek 59. IBM MQ bridge for HTTP

Webová aplikace IBM MQ bridge for HTTP přijímá požadavky HTTP od jednoho nebo více klientů. Zajišťuje interakci s produktem IBM MQ jejich jménem a vrací odpovědi HTTP na ně.

IBM MQ bridge for HTTP je servlet JEE , který je připojen k produktu IBM MQ pomocí adaptéru prostředků. Servlet HTTP zpracovává tři různé typy požadavků HTTP: **POST**, **GET**a **DELETE**.

Tabulka 84. IBM MQ bridge for HTTP sloves	
Požadavek HTTP	**Výsledek**
POST	Přepne zprávu do fronty nebo tématu.
GET	Prochází první zprávu ve frontě. V řádku s protokolem HTTP příkaz **GET** neodstraní zprávu z fronty. Nepoužívejte produkt **GET** s zasíláním zpráv publikování/odběru.
ODSTRANIT	Získá a odstraní zprávu z fronty nebo tématu.

Příklad HTTP POST

HTTP **POST** vloží zprávu do fronty nebo publikaci do tématu. Ukázka **HTTPPOST** Java je příkladem požadavku HTTP **POST** zprávy na frontu. Místo použití jazyka Java můžete vytvořit požadavek HTTP **POST** pomocí formuláře prohlížeče nebo sady nástrojů AJAX.

Následující obrázek ukazuje požadavek HTTP na vložení zprávy do fronty s názvem `myQueue`. Tento požadavek obsahuje záhlaví HTTP `x-msg-correlId`, které nastaví ID korelace zprávy IBM MQ.

```
POST /msg/queue/myQueue/ HTTP/1.1
Host: www.example.org
Content-Type: text/plain
x-msg-correlID: 1234567890
Content-Length: 50

Here is my message body that is posted on the queue.
```

*Obrázek 60. Příklad požadavku HTTP **POST** na frontu*

Následující obrázek ukazuje odezvu odeslanou zpět na klienta. Není žádný obsah odezvy.

```
HTTP/1.1 200 OK
Date: Wed, 2 Jan 2007 22:38:34 GMT
Server: Apache-Coyote/1.1 WMQ-HTTP/1.1 JEE-Bridge/1.1
Content-Length: 0
```

Obrázek 61. Příklad odezvy HTTP POST

Příklad HTTP DELETE

HTTP **DELETE** získá zprávu z fronty a odstraní ji, nebo načte a odstraní publikaci. Ukázka **HTTPDELETE** Java je příkladem požadavku HTTP **DELETE**, který čte zprávu z fronty. Místo použití jazyka Java můžete vytvořit požadavek HTTP **DELETE** pomocí formuláře prohlížeče nebo sady nástrojů AJAX.

Následující obrázek ukazuje požadavek HTTP, který odstraní následující zprávu z fronty s názvem `myQueue`. V odezvě se na klienta vrátí tělo zprávy. V terminologii IBM MQ je HTTP **DELETE** požadavek typu destructive get.

Požadavek obsahuje záhlaví požadavku HTTP `x-msg-wait`, které instruuje most IBM MQ pro HTTP, jak dlouho čekat na doručení zprávy do fronty. Požadavek dále obsahuje záhlaví požadavku `x-msg-require-headers`, které určuje, že klient má v odezvě přijmout ID korelace zprávy.

```
DELETE /msg/queue/myQueue/ HTTP/1.1
Host: www.example.org
x-msg-wait: 10
x-msg-require-headers: correlID
```

*Obrázek 62. Příklad požadavku HTTP **DELETE***

Následující obrázek ukazuje odezvu vrácenou na klienta. ID korelace se vrátí na klienta, jak požadovalo záhlaví požadavku `x-msg-require-headers`.

```
HTTP/1.1 200 OK
Date: Wed, 2 Jan 2007 22:38:34 GMT
Server: Apache-Coyote/1.1 WMQ-HTTP/1.1 JEE-Bridge/1.1
Content-Length: 50
Content-Type: text/plain; charset=utf-8
x-msg-correlId: 1234567890

Here is my message body that is retrieved from the queue.
```

*Obrázek 63. Příklad odezvy HTTP **DELETE***

Příklad HTTP GET

HTTP **GET** získá zprávu z fronty. Zpráva ve frontě zůstane. V rámci produktu IBM MQ je požadavek HTTP **GET** požadavkem na procházení. Požadavek HTTP **GET** můžete vytvořit pomocí klienta Java, formuláře prohlížeče nebo sady nástrojů AJAX.

Následující obrázek ukazuje požadavek HTTP na procházení další zprávou ve frontě s názvem `myQueue`.

Požadavek obsahuje záhlaví požadavku HTTP `x-msg-wait`, které instruuje IBM MQ bridge for HTTP, jak dlouho čekat na doručení zprávy do fronty. Požadavek dále obsahuje záhlaví požadavku `x-msg-require-headers`, které určuje, že klient má v odezvě přijmout ID korelace zprávy.

```
GET /msg/queue/myQueue/ HTTP/1.1
Host: www.example.org
x-msg-wait: 10
x-msg-require-headers: correlID
```

*Obrázek 64. Příklad požadavku HTTP **GET***

Následující obrázek ukazuje odezvu vrácenou na klienta. ID korelace se vrátí na klienta, jak požadovalo záhlaví požadavku `x-msg-require-headers`.

```
HTTP/1.1 200 OK
Date: Wed, 2 Jan 2007 22:38:34 GMT
Server: Apache-Coyote/1.1 WMQ-HTTP/1.1 JEE-Bridge/1.1
Content-Length: 50
Content-Type: text/plain; charset=utf-8
x-msg-correlId: 1234567890

Here is my message body that appears on the queue.
```

*Obrázek 65. Příklad odezvy HTTP **GET***

Instalace, konfigurace a ověření produktu IBM MQ bridge for HTTP

Získejte produkt IBM MQ bridge for HTTP instalací systému zpráv a webových služeb produktu Java z instalačního materiálu produktu IBM MQ MQI client nebo z instalačního materiálu serveru. Implementujte produkt IBM MQ bridge for HTTP na vhodný aplikační server.

Než začnete

Zkontrolujte předpokládané produkty na adrese Systémové požadavky pro IBM MQ. Proces instalace nekontroluje přítomnost a dostupnost předem vyžadovaného softwaru pro spuštění produktu IBM MQ bridge for HTTP. Musíte ověřit, zda jsou předpoklady nainstalovány.

IBM MQ bridge for HTTP je aplikace Java EE 4. Informace o podporovaných aplikačních serverech najdete v tématu Systémové požadavky pro IBM MQ.

Informace o této úloze

IBM MQ bridge for HTTP je dodáván jako soubor `.war` , `WMQHTTP.war`.

- V systémech UNIX a Linux:
  - `WMQHTTP.war` je součástí volby instalace systému zpráv Java a webových služeb. Tato volba je k dispozici v obou instalačních materiálech klienta i serveru.
  - Produkt `WMQHTTP.war` je nainstalován do produktu *mqmtop*`/java/http/WMQHTTP.war`. *mqmtop* je adresář, kde je nainstalován produkt IBM MQ .
  - Produkt `WMQHTTP.samples` je nainstalován do produktu *mqmtop*`/java/http/samples`. *mqmtop* je adresář, kde je nainstalován produkt IBM MQ .
- z/OS V systému z/OS:
  - `WMQHTTP.war` je součástí funkce komponenty IBM MQ z/OS UNIX System Services Components.
  - Produkt `WMQHTTP.war` je nainstalován do produktu *PathPrefix*`/usr/lpp/mqm/V7R0M0/HTTPBridge/`, kde *PathPrefix* je volitelná předpona definovaná zákazníkem.

Chcete-li instalovat produkt IBM MQ bridge for HTTP, implementovat a konfigurovat ho a ověřit konfiguraci, proveďte následující kroky instalace. Podrobnosti o krocích konfigurace se liší na různých aplikačních serverech. Použijte “Implementace a ověření produktu IBM MQ bridge for HTTP na WebSphere Application Server 6.1.0.9” na stránce 680 jako šablonu pro postup, jak postupovat na vašem aplikačním serveru.

Postup

1. Získejte produkt `WMQHTTP.war` instalací buď serveru IBM MQ MQI client , nebo serveru.
2. Zkopírujte produkt `WMQHTTP.war` na server, ze kterého jej lze implementovat na aplikační server.
3. Implementujte produkt `WMQHTTP.war` na aplikační server.
4. V případě potřeby nainstalujte produkt IBM MQ jako adaptér prostředků na aplikační server.

   Zjistěte, zda je produkt IBM MQ již nakonfigurován jako poskytovatel systému zpráv na aplikačním serveru. Použijte nástroj pro správu nebo správu dodaný s aplikačním serverem, abyste se podívali na IBM MQ. IBM MQ lze nalézt pod touto cestou, **Prostředky > JMS > Poskytovatelé systému zpráv**.
5. Konfigurovat továrnu připojení na aplikačním serveru pro připojení ke správci front, který používá přenos IBM MQ MQI client[7].
6. Nakonfigurujte webovou aplikaci `WMQHTTP.war` na aplikačním serveru, aby používala továrnu připojení.
7. Ověřte konfiguraci.

   a) Nastavte správce front uvedeného v továrně připojení a v lokální frontě.

[7] Nejprve nakonfigurujte přenos klienta alespoň na začátku. Některé aplikační servery se mohou připojit k produktu IBM MQ pomocí přímých připojení nebo připojení v režimu vázání.

b) Umístěte zprávu do lokální fronty.

c) Vytvořte kanál připojení serveru uvedený v továrně připojení s oprávněním pro čtení a zápis do lokální fronty.

d) Spusťte správce front a modul listener.

e) Spusťte aplikační server a server `WMQHTTP.war`, pokud ještě nejsou spuštěni.

f) Otevřete prohlížeč a zadejte příkaz `http://`*`hostname`*`:`*` web port`*`/`*`Context root`*`/msg/queue/`*`local queue`* .

Výsledky

V okně prohlížeče se zobrazí zpráva, kterou jste umístili do lokální fronty.

Jak pokračovat dále

1. Zkuste příklad, “Implementace a ověření produktu IBM MQ bridge for HTTP na WebSphere Application Server 6.1.0.9” na stránce 680.
2. Spusťte ukázkové aplikace HTTP Java .

Implementace a ověření produktu IBM MQ bridge for HTTP na WebSphere Application Server 6.1.0.9

Pomocí následujícího příkladu připravte implementaci produktu IBM MQ bridge for HTTP na spuštění ukázkových programů HTTP Java . Implementace je na WebSphere Application Server 6.1.0.9.

Než začnete

1. Postupujte podle pokynů v části “Instalace, konfigurace a ověření produktu IBM MQ bridge for HTTP” na stránce 679, chcete-li kopírovat produkt `WMQHTTP.war` na server přístupný pro vaši instalaci produktu WebSphere Application Server.
2. Nakonfigurujte správce front a frontu, aby bylo možné použít testování konfigurace:
   - V tomto příkladu je správce front konfigurován s použitím hodnot v produktu Tabulka 85 na stránce 680:

Tabulka 85. Konfigurace správce front	
Objekt	**Hodnota**
Název hostitele	`itso-01`
Správce front	`QM1`
Lokální fronta	`HTTPTESTQ`
Kanál připojení serveru	`MYSVRCON` Nakonfigurujte jméno uživatele MCA s dostatečným oprávněním pro čtení a zápis do `HTTPTESTQ`.
Port modulu listener	`1414`

3. Spustit správce front a modul listener
4. Umístěte testovací zprávu do produktu `HTTPTESTQ`. Příklad:
   a. Spusťte produkt IBM MQ Explorer.
   b. V seznamu lokálních front pro QM1klepněte pravým tlačítkem myši na volbu **HTTPTESTQ > Vložit testovací zprávu > typ `First Message` > Vložit zprávu > Zavřít** .
5. Spusťte aplikační server a přihlaste se do konzoly Integrated Solutions Console.

Informace o této úloze

Tento příklad ukazuje kroky, které je třeba provést, pokud spouštíte server WebSphere Application Server 6.1.0.9 jako aplikační server. Pokud provozujete jinou verzi produktu WebSphere Application Servernebo používáte jiný aplikační server, jsou kroky odlišné. Produkt WebSphere Application Server 6.1.0.9 je předkonfigurován s produktem IBM MQ instalovaném jako poskytovatel zpráv pomocí knihoven produktu IBM MQ MQI client . Není-li produkt IBM MQ předkonfigurován jako poskytovatel systému zpráv nebo pokud chcete použít vazby serveru IBM MQ , je třeba nainstalovat a nakonfigurovat adaptér prostředků produktu IBM MQ pro prostor JEE na aplikační server.

Postupujte podle pokynů k implementaci produktu IBM MQ bridge for HTTP na WebSphere Application Server 6.1.0.9a ověřte nasazení pomocí prohlížeče:

Postup

1. V navigačním podokně klepněte na volbu **Prostředky > Poskytovatelé JMS > Poskytovatel systému zpráv produktu IBM MQ**.

   Můžete nakonfigurovat buď na úrovni uzlu, buňky, nebo serveru, v závislosti na implementaci produktu WebSphere Application Server . V příkladu je použita implementace na úrovni serveru.
2. V části **Další vlastnosti**klepněte na volbu **Továrny na připojení > Nový**.
3. Ve formuláři poskytovatelů produktu JMS zadejte informace do produktu Tabulka 86 na stránce 681nebo alternativy, které vyberete, klepněte na tlačítko **Použít > Uložit**.

   Tabulka 86. Nastavit nebo upravit následující pole

Pole	**Hodnota**
Název	`WMQHTTPBridge`
Název rozhraní JNDI	`jms/WMQHTTPJCAConnectionFactory`
Správce front	`QM1`
Hostitel	`itso-01`
Port	`1414`
Kanál	`MYSVRCON`
Typ přenosu	`CLIENT`

4. V navigačním podokně klepněte na volbu **Aplikace > Instalovat novou aplikaci**.
5. Vložte cestu k produktu `WMQHTTP.war` do formuláře a zadejte kontextový kořenový adresář a klepněte na tlačítko **Další**.
   a) Kontextový kořenový adresář je volitelný. `mq` je výchozí kontextový kořenový adresář pro ukázkové aplikace HTTP.
   b) Kontextový kořenový adresář tvoří část identifikátoru URI identifikující produkt IBM MQ bridge for HTTP. Můžete vynechat kontextový kořenový adresář, nebo jej později změnit.
6. Na stránce **Vybrat volby instalace** průvodce instalací nemusíte měnit žádné výchozí nastavení, klepněte na tlačítko **Další**.
7. Na stránce **Mapovat moduly na servery** vyberte klastr nebo Server, zaškrtněte políčko Select a klepněte na tlačítko **Použít > Další**.
8. Na stránce **Mapovat odkazy na prostředky na prostředky** ve formuláři **javax.jms.ConnectionFactory** klepněte na tlačítko **Procházet ...** na řádku IBM MQ bridge for HTTP .
9. Na stránce **Podnikové aplikace > Dostupné prostředky** vyberte volbu **WMQHTTPBridge**a klepněte na tlačítko **Použít**.
10. Zpět ve formuláři **javax.jms.ConnectionFactory** vyberte metodu ověření.
    a) V případě příkladu vyberte volbu **Žádná**a klepněte na tlačítko **Použít**. Další volby vyžadují další konfiguraci.

11. Zaškrtněte zaškrtávací políčko **Vybrat** pro IBM MQ bridge for HTTP, klepněte na **Další > Další > Dokončit > Uložit**
12. V navigačním podokně klepněte na volbu **Aplikace > Podnikové aplikace**.
13. Zaškrtněte políčko pro výběr produktu `WMQHTTP.war` a klepněte na volbu **Spustit**.
14. Otevřete okno prohlížeče. Zadejte `http://itso-01:9080/mq/msg/queue/HTTPTEST` Qs použitím odpovídajícího názvu hostitele a portu.

Výsledky

Pokud je konfigurace úspěšná, zobrazí se okno prohlížeče `First Message`.

Jak pokračovat dále

Spusťte ukázkové aplikace HTTP Java .

Publikování/odběr pomocí produktu IBM MQ bridge for HTTP

Produkt IBM MQ bridge for HTTP používá rozhraní pro publikování a odběr produktu IBM MQ classes for JMS . HTTP **POST** vytvoří publikaci. Protokol HTTP **DELETE** vytvoří netrvalý spravovaný odběr. Před použitím identifikátoru URI tématu musíte nakonfigurovat publikování/odběr pro produkt JMS .

Publikování/odběr je plně integrován do produktu IBM WebSphere MQ 7 Před verzí 7. Samostatný zprostředkovatel publikování a odběru zpracoval publikace a odběry. Nazývá se "řazen do fronty" publikovat/odebírat, aby se odlišil od plně integrovaného publikování/odběru ve verzi 7. Produkt IBM WebSphere MQ 7 emuluje ve frontě publikování ve frontě s použitím integrovaného publikování/odběru. Emulace umožňuje, aby existující aplikace zařazené do fronty publikování/odběru existovaly společně s integrovanými aplikacemi spuštěnými ve stejném správci front. Aplikace typu publikování/odběr zařazené do fronty mohou také spolupracovat s integrovanými aplikacemi a sdílet stejná témata.
V produktu IBM WebSphere MQ 6byl zprostředkovatel dodáván s produktem IBM MQ; před tím, než byl produkt IBM WebSphere MQ 6 dostupný jako sada SupportPack.

Konfigurace

Produkt IBM MQ bridge for HTTP používá rozhraní produktu JMS k publikování a odběru. Ve verzi 7 můžete určit, zda má produkt IBM MQ classes for JMS používat ve frontě nebo integrované publikování/odběr pomocí vlastnosti `PROVIDERVERSION` JMS .

Další úvaha je, že můžete použít buď knihovny produktu IBM MQ MQI client s produktem IBM MQ bridge for HTTP, nebo knihovny serveru. Klientské knihovny produktu IBM WebSphere MQ 6 podporují pouze publikování/odběr ve frontě, zatímco podpora publikování a odběru ve frontě a integrovaného systému publikování/odběru verze 7. Většina webových nebo aplikačních serverů, které používají produkt IBM MQ jako poskytovatele systému zpráv, tak činí pomocí klientských knihoven. Chcete-li používat integrované publikování/odběr, musí být knihovny produktu IBM MQ MQI client i knihovny serveru alespoň ve verzi 7. Pokud je spuštěna dřívější verze produktu WebSphere než 7, musíte ve frontě publikovat publikování/subscribe; viz Tabulka 87 na stránce 682. Zkontrolujte, které knihovny jsou nainstalovány nebo nakonfigurovány s webovým serverem nebo aplikačním serverem, který používáte.

Tabulka 87. Režimy konfigurace publikování a odběru

	Klient V6 nebo starší	**Klient V7 nebo pozdější**
Server V6 nebo starší	1. Spusťte skript `\java\bin\MQJMS_PSQ.mqsc` .	Nepodporováno

Tabulka 87. Režimy konfigurace publikování a odběru (pokračování)		
	Klient V6 nebo starší	**Klient V7 nebo pozdější**
Server V7 nebo pozdější	1. Spusťte skript `\java\bin\MQJMS_PSQ.mqsc` . 2. Nastavit správce front na `PSMODE=ENABLED`	1. Pokud PROVIDERVERSION = 7 a. Nastavit správce front na `PSMODE=ENABLED` nebo `PSMODE=COMPAT` 2. Pokud PROVIDERVERSION = 6 a. Nastavit správce front na `PSMODE=ENABLED`

Publikovat

Odešlete požadavek HTTP **POST** s identifikátorem URI:

```
http://hostname: port/context_root/msg/topic/topicString
```

Obsahy zpráv se publikují za použití řetězce tématu *`topicString`*.

Odebírat

Odešlete požadavek HTTP **DELETE** s identifikátorem URI:

```
http://hostname: port/context_root/msg/topic/topicString
```

Produkt IBM MQ bridge for HTTP vytvoří spravovaný netrvalý odběr pro řetězec tématu *`topicString`*. Odběr je odstraněn, jakmile je vrácena publikace, nebo do vypršení intervalu čekání nastaveného vlastním záhlavím entity `x-msg-wait`.

Spuštění ukázek produktu IBM MQ bridge for HTTP

Ukázky IBM MQ bridge for HTTP jsou k dispozici pouze pro použití v operačním systému Windows . The samples show you how to submit HTTP **POST** and HTTP **DELETE** commands to IBM MQ bridge for HTTP from Java programs.

Než začnete

Ověřte svůj most IBM MQ pro instalaci HTTP spuštěním kroku “7” na stránce 679 v publikaci “Instalace, konfigurace a ověření produktu IBM MQ bridge for HTTP” na stránce 679.

Ukázky HTTP se nainstalují do adresářů zobrazených v Tabulka 88 na stránce 683. V každém případě je zdrojový kód nainstalován do podadresáře `/src` .

Tabulka 88. Umístění ukázek HTTP	
Platforma	**Umístění**
Windows	*`MQ_INSTALLATION_PATH`*`/tools/http/samples`
z/OS	*`PathPrefix`*`/usr/lpp/mqm/V7R0M0/http/samples`
IBM i	*`MQ_INSTALLATION_PATH`*`/java/samples/http`

Tabulka 88. Umístění ukázek HTTP (pokračování)	
Platforma	**Umístění**
Všechny ostatní platformy	*MQ_INSTALLATION_PATH*/samp/http

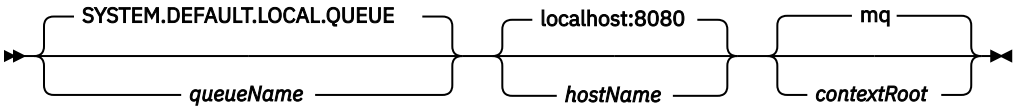
MQ_INSTALLATION_PATH představuje adresář, kde je nainstalován produkt IBM MQ .

Informace o této úloze

Ukázky simulují ukázkové aplikace IBM MQ AMQSPUT a AMQSGET. Vykreslují následující funkce v prostředí systému zpráv typu point-to-point:

- **HTTPPOST** -Odesílá požadavky HTTP **POST** v aplikaci Java za účelem vložení zpráv do fronty IBM MQ pomocí IBM MQ bridge for HTTP a zpracovává odpovědi.
- **HTTPDELETE** -Odesílá požadavky HTTP **DELETE** v aplikaci Java za účelem získání zpráv z fronty IBM MQ pomocí IBM MQ bridge for HTTP a zpracovává odpovědi, které obsahují zprávu IBM MQ .

Parametry pro HTTPPOST a HTTPDELETE



Chcete-li spustit ukázku **HTTPPOST** , proveďte následující kroky:

Postup

1. V příkazovém okně přejděte do adresáře ukázek HTTP.
2. Spusťte ukázku **HTTPPOST** .

   ```
   java -classpath . HTTPPOST [parameters]
   ```

   Když se spustí ukázka **HTTPPOST** , zobrazí se následující výstup:

   ```
   HTTP POST Sample start
   Target server is ' hostName '
   Target queue is ' queueName '
   Target context-root is ' contextRoot '
   ```

3. Do příkazového řádku zadejte text, který má tvořit tělo zprávy.
4. Stisknutím klávesy Enter zveřejněte zprávu do fronty produktu IBM MQ .
   a) Chcete-li odeslat další zprávu, zadejte nějaký další text.
      Text tvoří tělo druhé zprávy IBM MQ .
   b) Stisknutím klávesy Enter zveřejněte zprávu do fronty produktu IBM MQ .
5. Stiskněte dvakrát klávesu Enter, abyste ukončili **HTTPPOST**.
   Zobrazí se následující výstup:

   ```
   HTTP POST Sample end
   ```

Jak pokračovat dále

Ukázka **HTTPDELETE** provádí destruktivní získání všech zpráv, které jste umístili do fronty IBM MQ .

Spusťte ukázku produktu **HTTPDELETE** provedením následujících kroků:

1. V okně příkazového řádku přejděte na *MQ_INSTALLATION_PATH*/tools/samples.
   MQ_INSTALLATION_PATH představuje adresář, kde je nainstalován produkt IBM MQ .

2. Spusťte ukázku **HTTPDELETE** .

```
java -classpath . HTTPPOST [parameters]
```

Když se spustí ukázka **HTTPDELETE** , zobrazí se následující výstup:

```
HTTP DELETE Sample start
Target server is ' host:port '
Target queue is ' your queue name '
Target context-root is ' your context-root '
message
message
...
```

Aspekty zabezpečení pro produkt IBM MQ bridge for HTTP

Pro ověření klienta webového prohlížeče platí standardní aspekty zabezpečení webu. Oprávnění k prostředkům produktu IBM MQ je na úrovni uživatele, který spouští servlet IBM MQ bridge for HTTP , a nikoli jednotlivé klienty webového prohlížeče. Standardní bezpečnostní protihodnota IBM MQ se vztahuje na IBM MQ.

Data proudící z webového prohlížeče do aplikace produktu IBM MQ pomocí produktu IBM MQ bridge for HTTPa zpět, trvá tři kroky:

Připojení klienta
Z prohlížeče do produktu IBM MQ bridge for HTTP prostřednictvím protokolu TCP/IP pomocí protokolu HTTP.

Připojení adaptéru prostředků k produktu IBM MQ
Připojení je z produktu IBM MQ bridge for HTTP ke správci front produktu IBM MQ . Připojení je buď připojení klienta, přes TCP/IP, nebo lokální připojení vazeb IBM MQ . Po navázání připojení je požadavek HTTP umístěn na standardní lokální frontu nebo na přenosovou frontu.

Z lokální fronty produktu IBM MQ přes jeden nebo více kanálů do cílové fronty.
Použití standardních technik pro zabezpečení front, témat, správců front a kanálů.

Odpověď podnivá kroky opačně.

Připojení klienta

Zabezpečená připojení mezi klienty HTTP a aplikačním serverem pomocí webového kontejneru. Použít standardní metody serveru HTTP, jako např. použití HTTPS. Informace naleznete v dokumentaci k aplikačnímu serveru.

Připojení adaptéru prostředků k produktu IBM MQ

Připojení mezi adaptérem prostředků a správcem front je autorizováno pouze s použitím jednoho ID uživatele. Přiřaďte jedno ID uživatele k identifikaci požadavků z produktu IBM MQ bridge for HTTP. ID uživatele musí mít omezené autorizace IBM MQ pouze pro ty externí uživatele, kteří musí mít přístup. Musíte ověřit skutečného klienta samostatně a navázat důvěru pro následné interakce s klientem pomocí standardních technik pro webové zabezpečení.

Zabezpečte připojení mezi adaptérem prostředků a správcem front pomocí jediného ID uživatele. Omezte oprávnění, které ID uživatele nemá, aby více než bylo potřeba ke čtení a zápisu zpráv do front a témat. IBM MQ bridge for HTTP je bod napadení mezi internetem a intranet.

Způsob zabezpečení spojení mezi adaptérem prostředků a produktem IBM MQ je závislý na konkrétním adaptéru prostředků. Další informace naleznete v dokumentaci k adaptéru prostředků.

Vyvíjení aplikací MQI s produktem IBM MQ

Produkt IBM MQ poskytuje podporu pro jazyky C, Visual Basic, COBOL, Assembler, RPG, pTALa PL/I. Tyto procedurální jazyky používají rozhraní fronty zpráv (MQI) pro přístup ke službám front zpráv.

Podrobné informace o tom, jak psát vaše aplikace ve zvoleném jazyce, najdete v dílčích tématech.

Přehled rozhraní volání pro procedurální jazyky naleznete v tématu Popisy volání. Toto téma obsahuje seznam volání MQI a každé volání ukazuje, jak kódovat volání v každém z těchto jazyků.

Produkt IBM MQ poskytuje soubory definic dat, které vám pomohou psát vaše aplikace. Úplný popis viz "Soubory definic dat produktu IBM MQ" na stránce 686.

Jako pomoc při výběru procedurálního jazyka pro programování vašich programů zvažte maximální délku zpráv, které budou programy zpracovávat. Pokud budou vaše programy zpracovávat pouze zprávy o známé maximální délce, můžete je kódovat v kterémkoli z podporovaných jazyků. Pokud neznáte maximální délku zpráv, které budou programy muset zpracovat, bude vámi zvolený jazyk záviset na tom, zda zapisujete CICS, IMSnebo dávkovou aplikaci:

IMS a dávka
Kódujte programy v jazycích C, PL/I nebo assembler tak, aby používaly funkce těchto jazyků k získání a uvolnění libovolných množství paměti. Alternativně můžete kódovat své programy v jazyce COBOL, ale používat podprogramy jazyka v jazyku assembler, PL/I nebo C k získání a uvolnění paměti.

CICS
Kódujte programy v libovolném jazyce, který je podporován produktem CICS. Rozhraní EXEC CICS poskytuje volání pro správu paměti, je-li to nutné.

Související pojmy

"Objektově orientované aplikace" na stránce 10
Produkt IBM MQ poskytuje podporu pro produkty .NET, ActiveX, C + +, Javaa JMS. Tyto jazyky a rámce používají objektový model produktu IBM MQ , který poskytuje třídy, které poskytují stejné funkce jako volání a struktury produktu IBM MQ . Některé z jazyků a rámců, které používají model objektů produktu IBM MQ , poskytují další funkce, které nejsou k dispozici při použití procedurálních jazyků s rozhraním MQI (Message Queue Interface).

"Koncepty vývoje aplikací" na stránce 6
K zápisu aplikací IBM MQ můžete použít volbu procedurálních nebo objektově orientovaných jazyků. Odkazy v tomto tématu použijte pro informace o koncepcích produktu IBM MQ , které jsou užitečné pro vývojáře aplikací.

Související informace

Technický přehled
Odkaz na vývoj aplikací

Soubory definic dat produktu IBM MQ

Produkt IBM MQ poskytuje soubory definic dat, které vám pomohou psát vaše aplikace.

Soubory definice dat jsou také známé jako:

Jazyk	Definice dat
C	Zahrnout soubory nebo hlavičkové soubory
Visual Basic	Soubory modulu (pouze 32bitové verze)
COBOL	Kopírovat soubory
Asembler	Makra
PL/I	Zahrnout soubory

Soubory definic dat, které vám pomohou zapsat uživatelské procedury, jsou popsány v souboru IBM MQ COPY, header, include a module files.

Soubory definic dat, které vám pomohou s procedurami pro psaní instalovatelných služeb, jsou popsány v tématu "Uživatelské procedury, uživatelské procedury rozhraní API a instalovatelné služby produktu IBM MQ" na stránce 897.

Informace o souborech definic dat podporovaných na C + + viz Použití C++.

IBM i

Informace o souborech definic dat podporovaných v RPG najdete v příručce IBM i Application Programming Reference (ILE/RPG).

Názvy souborů definic dat mají předponu CMQ a příponu, která je určena programovacím jazykem:

Přípona	Jazyk
a	Jazyk asembleru
b	Visual Basic
c	C
l	COBOL (bez inicializovaných hodnot)
p	PL/I
v	COBOL (s výchozí sadou hodnot)

Instalační knihovna

Název **thlqual** je kvalifikátor vyšší úrovně instalační knihovny v systému z/OS.

Toto téma uvádí soubory definic dat produktu IBM MQ , pod těmito nadpisy:

- “Zahrnují soubory jazyka C” na stránce 687
- “Soubory modulu Visual Basic” na stránce 687
- “Soubory kopie COBOL” na stránce 688
- z/OS “Makra v assembleru System/390” na stránce 689
- z/OS “Soubory začlenění PL/I” na stránce 689

Zahrnují soubory jazyka C

Soubory začlenění IBM MQ C jsou vypsány v hlavičkovém souboru C. Jsou instalovány v následujících adresářích nebo knihovnách:

Platforma	Instalační adresář nebo knihovna
IBM i	QMQM/H
IBM i IBM i	
UNIX	*MQ_INSTALLATION_PATH*/inc/
Systémy Windows	*MQ_INSTALLATION_PATH*\Tools\c\include
z/OS	**thlqual**.SCSQC370
z/OS z/OS	

kde *MQ_INSTALLATION_PATH* představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Poznámka: Pro UNIXjsou soubory začlenění symbolicky propojeny do `/usr/include`.

Další informace o struktuře adresářů najdete v tématu Plánování podpory systému souborů.

Soubory modulu Visual Basic

Produkt IBM MQ for Windows poskytuje čtyři soubory modulu Visual Basic.

Jsou uvedeny v části Soubory modulu Visual Basic a instalovány v

```
MQ_INSTALLATION_PATH\Tools\Samples\VB\Include
```

Soubory kopie COBOL

Pro COBOL poskytuje produkt IBM MQ samostatné soubory kopií obsahující pojmenované konstanty a dva soubory kopií pro každou ze struktur.

Pro každou strukturu existují dva kopírovací soubory, protože každý je poskytován jak s počátečními hodnotami, tak bez počátečních hodnot:

- Ve funkci WORKING-STORAGE SECTION programu COBOL použijte soubory, které inicializují pole struktury na výchozí hodnoty. Tyto struktury jsou definovány v kopírovaných souborech, které mají příponu s příponou V (hodnoty).
- V oddíle LINKAGE v programu COBOL použijte struktury bez počátečních hodnot. Tyto struktury jsou definovány ve kopírovacích souborech, které mají příponu s písmenem L (páková).

IBM i Kopíruji soubory obsahující data a definice rozhraní pro produkt IBM i jsou poskytovány pro programy ILE COBOL pomocí prototypovaných volání do rozhraní MQI. Soubory existují v QMQM/QCBLLESRC s názvy členů, které mají příponu L (pro struktury bez počátečních hodnot), nebo příponu V (pro struktury s počátečními hodnotami).

Soubory kopie produktu IBM MQ COBOL jsou uvedeny v seznamu Soubory COBOL COPY. Jsou instalovány v následujících adresářích:

Platforma	Instalační adresář nebo knihovna
Ostatní platformy UNIX	*MQ_INSTALLATION_PATH*/inc/
IBM i IBM i IBM i	QMQM/QCBLLESRC
Windows	*MQ_INSTALLATION_PATH*\Tools\cobol\copybook (for Micro Focus COBOL) *MQ_INSTALLATION_PATH*\Tools\cobol\copybook\VAcobol (pro produkt IBM VisualAge COBOL)
z/OS z/OS z/OS	**thlqual**.SCSQCOBC

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Do svého programu zahrňte pouze ty soubory, které potřebujete. Postupujte takto s jedním nebo více příkazy COPY po deklaraci úrovně level-01 . To znamená, že v případě potřeby můžete zahrnout více verzí struktur do programu. Všimněte si, že CMQV je velký soubor.

Zde je příklad kódu COBOL pro zahrnutí souboru kopie CMQMDV:

```
01 MQM-MESSAGE-DESCRIPTOR.
COPY CMQMDV.
```

Každá deklarace struktury začíná položkou level-01 ; můžete deklarovat několik instancí struktury zakódováním deklarace level-01 , za kterou následuje příkaz COPY, který se kopíruje ve zbytku deklarace struktury. Chcete-li se odkázat na odpovídající instanci, použijte klíčové slovo IN.

Zde je příklad kódu COBOL pro zahrnutí dvou instancí CMQMDV:

```
* Declare two instances of MQMD
01 MY-CMQMD.
COPY CMQMDV.
01 MY-OTHER-CMQMD.
COPY CMQMDV.
*
```

```
* Set MSGTYPE field in MY-OTHER-CMQMD
MOVE MQMT-REQUEST TO MQMD-MSGTYPE IN MY-OTHER-CMQMD.
```

Srovnejte struktury na 4bajtových hranicím. Použijete-li příkaz COPY k zahrnutí struktury za položkou, která není položkou level-01 , ujistěte se, že struktura je násobkem 4-bajtů od začátku položky level-01 . Pokud to neuděláte, můžete snížit výkon aplikace.

Struktury jsou popsány v části Datové typy použité v rozhraní MQI. Popisy polí ve strukturách zobrazují názvy polí bez předpony. V programech COBOL, prefix názvů polí s názvem struktury následované pomlčkou, jak je uvedeno v deklaracích COBOL. Pole v souborech kopií struktury se tímto způsobem připojí jako předpona.

Názvy polí v deklaracích v souborech kopií struktury jsou velkými písmeny. Místo toho můžete použít malá i velká písmena. Například pole *StrucId* struktury MQGMO je zobrazeno jako MQGMO-STRUCID v deklaraci COBOL a v souboru kopie.

Struktury V-suffix jsou deklarovány s počátečními hodnotami pro všechna pole, takže je třeba nastavit pouze ta pole, kde se požadovaná hodnota liší od počáteční hodnoty.

Makra v assembleru System/390

z/OS

Produkt IBM MQ for z/OS poskytuje dvě makra v asembleru, která obsahují pojmenované konstanty, a jedno makro pro generování každé struktury.

Jsou uvedeny v souboru z/OS Assembler COPY files a jsou instalovány v **thlqual**.SCSQMACS.

Tato makra jsou volána pomocí kódu, jako je tento:

```
MY_MQMD CMQMDA EXPIRY=0,MSGTYPE=MQMT_DATAGRAM
```

Soubory začlenění PL/I

z/OS

Produkt IBM MQ for z/OS poskytuje soubory obsahující všechny definice, které potřebujete při zápisu aplikací IBM MQ do PL/I.

Soubory jsou vypsány v souborech začlenění PL/I a jsou nainstalovány v adresáři **thlqual**.SCSQPLIC:

Zahrňte tyto soubory do svého programu, pokud se chystáte propojit stub IBM MQ s vaším programem (viz “Příprava programu na spuštění” na stránce 1000). Zahrnujte pouze CMQP, pokud chcete dynamicky propojit volání IBM MQ (viz “Dynamické volání stubu IBM MQ” na stránce 1007). Dynamické propojení lze provádět pouze pro dávkové programy a programy IMS .

Psaní procedurální aplikace pro řazení do fronty

Prostřednictvím těchto informací můžete získat informace o vytváření aplikací pro řazení do front, připojování a odpojování od správce front, publikování a odběru a otevírání a zavírání objektů.

Následující odkazy použijte k vyhledání více informací o psaní aplikací:

- “Přehled rozhraní fronty zpráv” na stránce 690
- “Připojování k správci front a odpojování od něj” na stránce 704
- “Otevírání a zavírání objektů” na stránce 712
- “Vložení zpráv do fronty” na stránce 723
- “Získávání zpráv z fronty” na stránce 737
- “Zapisování aplikací typu publikování/odběr” na stránce 774
- “Inquaning about a nastavení atributů objektu” na stránce 815
- “Potvrzení a zálohování jednotek práce” na stránce 818

- “Spuštění aplikací produktu IBM MQ pomocí spouštěčů” na stránce 829
- “Práce s MQI a klastry” na stránce 847
- z/OS “Použití a zápis aplikací v systému IBM MQ for z/OS” na stránce 851
- z/OS “Aplikace mostu IMS a IMS v systému IBM MQ for z/OS” na stránce 57

Související pojmy

“Koncepty vývoje aplikací” na stránce 6
K zápisu aplikací IBM MQ můžete použít volbu procedurálních nebo objektově orientovaných jazyků. Odkazy v tomto tématu použijte pro informace o koncepcích produktu IBM MQ , které jsou užitečné pro vývojáře aplikací.

“Vyvíjení aplikací pro IBM MQ” na stránce 5
Můžete vyvíjet aplikace k odesílání a přijímání zpráv a ke správě správců front a souvisejících prostředků. IBM MQ podporuje aplikace napsané v mnoha různých jazycích a rámcích.

“Aspekty návrhu pro aplikace produktu IBM MQ” na stránce 43
Když se rozhodnete, jak vaše aplikace mohou využívat výhody platforem a prostředí, které máte k dispozici, musíte se rozhodnout, jak používat funkce nabízené produktem IBM MQ.

“Zápis procedurálních aplikací klienta” na stránce 874
Co potřebujete vědět, chcete-li psát klientské aplikace na serveru IBM MQ pomocí procedurálního jazyka.

“Sestavení procedurální aplikace” na stránce 966
Aplikaci IBM MQ můžete psát v jednom z několika procedurálních jazyků a aplikaci spustit na několika různých platformách.

“Obsluha chyb procedurálních programů” na stránce 1015
Tyto informace vysvětlují chyby související s voláními MQI MQI buď při volání, nebo při doručení její zprávy do konečného cíle.

Související úlohy

“Použití ukázkových procedurálních programů produktu IBM MQ” na stránce 1034
Tyto ukázkové programy jsou napsány v procedurálních jazycích a demonstrují typická použití rozhraní MQI (Message Queue Interface). Programy produktu IBM MQ na různých platformách.

“Vývoj webových služeb pomocí produktu IBM MQ” na stránce 1255
Můžete vyvíjet aplikace produktu IBM MQ pro webové služby pomocí přenosu IBM MQ pro protokol SOAP.

Přehled rozhraní fronty zpráv

Zde jsou uvedeny informace o komponentách rozhraní MQI (Message Queue Interface).

Rozhraní fronty zpráv se skládá z následujících prvků:

- *Volání* , jejichž prostřednictvím mohou programy přistupovat ke správci front a k jeho zařízením.
- *Struktury* , které programy používají k předávání dat do správce front a získávání dat z těchto správců front.
- *Elementární datové typy* pro předávání dat do správce front a získávání dat ze správce front

z/OS IBM MQ for z/OS také dodává:

- Dvě další volání, pomocí kterých mohou dávkové programy z/OS potvrdit a vrátit změny.
- *Soubory definice dat* (někdy známé jako kopírovací soubory, makra, soubory začlenění a hlavičkové soubory), které definují hodnoty konstant dodávaných s IBM MQ for z/OS.
- *Stub programy* pro propojení-úpravy s vašimi aplikacemi.
- Sada ukázkových programů, které demonstrují, jak používat rozhraní MQI na platformě z/OS . Další informace o těchto ukázkách naleznete v tématu “Použití ukázkových programů pro produkt z/OS” na stránce 1137.

IBM i IBM MQ for IBM i také dodává:

- *Soubory definice dat* (někdy známé jako kopírovací soubory, makra, soubory začlenění a hlavičkové soubory), které definují hodnoty konstant dodávaných s IBM MQ for IBM i.
- Tři programy typu stub pro linkování-úpravy na aplikace ILE C, ILE COBOL a ILE RPG.
- Sada ukázkových programů, které demonstrují, jak používat rozhraní MQI na platformě IBM i .

IBM MQ for Windows a IBM MQ na systémech UNIX and Linux poskytují také:

- Volání, která umožňuje IBM MQ for Windows a IBM MQ v systémových programech UNIX and Linux může potvrdit a zazálohovat změny.
- *Zahrnout soubory* , které definují hodnoty konstant dodávaných na těchto platformách.
- *Soubory knihovny* pro propojení vašich aplikací.
- Sada ukázkových programů, které demonstrují, jak používat rozhraní MQI na těchto platformách. Další informace o těchto ukázkách naleznete v tématu “Použití ukázkových programů na více platformách” na stránce 1035.
- Ukázka zdrojového kódu a spustitelného kódu pro vazby na externí správce transakcí.

Chcete-li zjistit více o rozhraní MQI, použijte následující odkazy:

- “Volání MQI” na stránce 692
- “Volání synchronizačního bodu” na stránce 692
- “Převod dat, datové typy, definice dat a struktury” na stránce 693
- “Soubory typu stub a soubory knihovny produktu IBM MQ” na stránce 694
- “Parametry společné pro všechna volání” na stránce 700
- “Určení vyrovnávacích pamětí” na stránce 701
- z/OS “Aspekty dávky produktu z/OS” na stránce 701
- “Zpracování signálů UNIX and Linux” na stránce 702

Související pojmy

“Připojování k správci front a odpojování od něj” na stránce 704
Chcete-li používat programovací služby IBM MQ , musí mít program připojení ke správci front. V této části se dozvíte, jak se připojit k správci front a odpojit je od správce front.

“Otevírání a zavírání objektů” na stránce 712
Tyto informace poskytují vhled do otevírání a zavírání objektů IBM MQ .

“Vložení zpráv do fronty” na stránce 723
V této části se dozvíte, jak vložit zprávy do fronty.

“Získávání zpráv z fronty” na stránce 737
Tyto informace použijte k získání informací o získávání zpráv z fronty.

“Inquaning about a nastavení atributů objektu” na stránce 815
Atributy jsou vlastnosti, které definují charakteristiky objektu IBM MQ .

“Potvrzení a zálohování jednotek práce” na stránce 818
Tyto informace popisují, jak potvrdit a zazálohovat všechny zotavitelné operace get a put, které se vyskytly v jednotce práce.

“Spuštění aplikací produktu IBM MQ pomocí spouštěčů” na stránce 829
Informace o spouštěčích a o tom, jak spustit aplikace IBM MQ pomocí spouštěčů.

“Práce s MQI a klastry” na stránce 847
Existují speciální volby na voláních a návratových kódech, které se vztahují ke klastrování.

“Použití a zápis aplikací v systému IBM MQ for z/OS” na stránce 851
Aplikace produktu IBM MQ for z/OS mohou být vytvořeny z programů spuštěných v mnoha různých prostředích. To znamená, že mohou využít výhody zařízení dostupných ve více než jednom prostředí.

“Aplikace mostu IMS a IMS v systému IBM MQ for z/OS” na stránce 57
Tyto informace vám pomohou při psaní aplikací produktu IMS pomocí produktu IBM MQ.

Volání MQI

Tyto informace použijte k seznámení se s voláními v rozhraní MQI (Message Queue Interface).

Volání modulu MQI lze seskupit podle následujících kroků:

MQCONN, MQCONNX a MQDISC
Pomocí těchto volání můžete připojit program k (s volbami nebo bez voleb) a odpojit program od správce front. Pokud zapisujete programy CICS pro produkt z/OS, nemusíte tato volání používat. Doporučuje se je však použít, chcete-li aplikaci portovat na jiné platformy.

MQOPEN a MQCLOSE
Pomocí těchto volání můžete otevřít a zavřít objekt, jako je například fronta.

MQPUT a MQPUT1
Použijte tato volání k vložení zprávy do fronty.

MQGET
Toto volání můžete použít k procházení zpráv ve frontě nebo k odebírání zpráv z fronty.

MQSUB, MQSUBRQ
Pomocí těchto volání zaregistrujte odběr do tématu a vyžádejte publikování odpovídající odběru.

MQINQ
Použijte toto volání k dotazům na atributy objektu.

MQSET
Použijte toto volání k nastavení některých atributů fronty. Atributy jiných typů objektů nelze nastavit.

MQBEGIN, MQCMIT a MQBACK
Použijte tato volání, je-li IBM MQ koordinátorem jednotky práce. Funkce MQBEGIN spustí jednotku práce. MQCMIT a MQBACK ukončí pracovní jednotku, a to buď potvrdit, nebo odvolávání aktualizací provedených během transakce. IBM i Produkt IBM i se používá ke koordinaci globálních transakcí na serveru IBM MQ for IBM i. Jsou použity příkazy vázaného zpracování vázaného zpracování, vázaného zpracování a odvolání.

MQCRTMH, MQBUFMH, MQMBUBUF, MQDLTMH
S použitím těchto volání můžete vytvořit popisovač zprávy, převést popisovač zprávy na vyrovnávací paměť nebo vyrovnávací paměť na popisovač zprávy a odstranit popisovač zprávy.

MQSETTMP, MQINQMP, MQDLTMP
Pomocí těchto volání můžete nastavit vlastnost zprávy na popisovači zprávy, zjišťovat vlastnost zprávy a odstranit vlastnost z manipulátoru zprávy.

MQCB, MQCB_FUNCTION, MQCTL
Tato volání slouží k registraci a řízení funkce zpětného volání.

MQSTAT
Použijte toto volání k načtení informací o stavu pro předchozí asynchronní operace put.

Popis volání MQI naleznete v tématu Popisy volání .

Volání synchronizačního bodu

Tyto informace použijte k vyhledání informací o volání synchronizačního bodu na různých platformách.

Volání bodu synchronizace jsou k dispozici následujícím způsobem:

IBM MQ for z/OS volání

z/OS

Produkt IBM MQ for z/OS poskytuje volání MQCMIT a MQBACK.

Pomocí těchto volání v dávkových programech produktu z/OS můžete správci front sdělit, že všechny operace MQGET a MQPUT od posledního bodu synchronizace mají být trvalé (potvrzené) nebo mají být vráceny zpět. Chcete-li potvrdit a vrátit změny v jiných prostředích, postupujte takto:

CICS
Použijte příkazy jako EXEC CICS SYNCPOINT a EXEC CICS SYNCPOINT ROLLBACK.

IMS
Použijte zařízení pro synchronizaci bodu IMS , jako např. GU (get unique) na volání IOPCB, CHKP (checkpoint) a ROLB (rollback).

RRS
Podle potřeby použijte MQCMIT a MQBACK nebo SRRCMIT a SRRBACK. (Viz “Správa transakcí a zotavitelné služby správce prostředků” na stránce 822.)

Poznámka: Atributy SRRCMIT a SRRBACK jsou nativní příkazy RRS, nejedná se o volání MQI.

IBM i volání

IBM i

Produkt IBM MQ for IBM i poskytuje příkazy MQCMIT a MQBACK. Můžete také použít příkazy IBM i COMMIT a ROLLBACK nebo jakékoli jiné příkazy nebo volání, které iniciují zařízení pro vázané zpracování IBM i (například EXEC CICS SYNCPOINT).

Volání IBM MQ na platformách Windows, UNIX and Linux

ULW

Následující produkty poskytují volání MQCMIT a MQBACK:

- IBM MQ for Windows
- IBM MQ v systémech UNIX and Linux

Pomocí volání synchronizačních bodů v programech můžete sdělit správci front, že všechny operace MQGET a MQPUT od posledního bodu synchronizace mají být trvale provedeny (potvrzeny) nebo mají být vráceny zpět. Chcete-li potvrdit a zazálohovat změny v prostředí CICS , použijte příkazy jako EXEC CICS SYNCPOINT a EXEC CICS SYNCPOINT ROLLBACK.

Převod dat, datové typy, definice dat a struktury

Tyto informace použijte, chcete-li se dozvědět více o konverzích dat, základních datových typech, definicích dat IBM MQ a strukturách při použití rozhraní fronty zpráv.

Převod dat

Volání MQXCNVC (převod znaků) převádí znaková data zprávy z jednoho znaku na jiný. S výjimkou systému IBM MQ for z/OSse toto volání používá pouze z uživatelské procedury pro převod dat.

Viz MQXCNVC-Konvertování znaků pro syntaxi použitou při volání MQXCNVC a “Zápis uživatelských procedur pro převod dat” na stránce 948 pro navádění na psaní a vyvolání uživatelských procedur pro převod dat.

Elementární datové typy

Pro podporované programovací jazyky poskytuje rozhraní MQI elementární datové typy nebo nestrukturovaná pole.

Tyto datové typy jsou plně popsány v části Elementární datové typy.

IBM MQ Definice dat

z/OS Produkt IBM MQ for z/OS dodává definice dat ve formě souborů kopie jazyka COBOL, maker pro sestavení pro sestavení, jediného souboru začlenění jazyka PL/I, jazyka obecného jazyka C a souborů začlenění jazyka C + +.

IBM i Produkt IBM MQ for IBM i dodává definice dat ve formě souborů kopií v jazyce COBOL, kopírovacích souborů RPG, souborů jazyka C včetně souborů jazyka C a souborů jazyka C + +.

Soubory definic dat dodávané s produktem IBM MQ obsahují:

- Definice všech konstant IBM MQ a návratových kódů

- Definice struktur a datových typů produktu IBM MQ
- Konstantní definice pro inicializaci struktur
- Prototypy funkce pro každý z volání (pouze pro jazyk PL/I a jazyk C)

Úplný popis definičních souborů dat produktu IBM MQ naleznete v tématu “Soubory definic dat produktu IBM MQ” na stránce 686.

Struktury

Struktury, které jsou použity s voláními MQI uvedenými v produktu “Volání MQI” na stránce 692, jsou dodávány v souborech definice dat pro každý podporovaný programovací jazyk.

IBM i z/OS Soubory IBM MQ for z/OS a IBM MQ for IBM i obsahují konstanty, které je třeba použít při vyplňování některých polí těchto struktur. Další informace o těchto zdrojích naleznete v tématu Definice dat produktu IBM MQ.

Souhrn struktur najdete v tématu Souhrn datových typů struktury .

Soubory typu stub a soubory knihovny produktu IBM MQ

Zde jsou uvedeny programy stubu a soubory knihoven, které jsou zde uvedeny, pro každou platformu.

Další informace o tom, jak používat programy typu stub a soubory knihoven při vytváření spustitelné aplikace, najdete v tématu “Sestavení procedurální aplikace” na stránce 966. Informace o odkazech na soubory knihovny C + +, viz Použití C++ *IBM MQ Použití C++*.

AIX *IBM MQ for AIX*

V systému IBM MQ for AIXmusíte propojit svůj program se soubory knihovny MQI dodanými pro prostředí, ve kterém spouštíte aplikaci, kromě těch, které poskytuje operační systém.

V aplikaci bez podprocesů se připojte k jedné z následujících knihoven:

Tabulka 89. Soubory knihovny pro aplikace bez podprocesů AIX

Soubor knihovny	**Prostředí**
libmqm.a	Server pro C
libmqic.a & **libmqm.a**	Klient pro C
libmqmzf.a	Výstupy instalovatelné služby pro C
libmqmxa.a	Rozhraní XA serveru
libmqmxa64.a	Alternativní rozhraní XA serveru
libmqcxa.a	Rozhraní XA klienta
libmqcxa64.a	Alternativní rozhraní XA klienta
libmqmcbrt.o	Běhová knihovna produktu IBM MQ pro podporu Micro Focus COBOL
libmqmcb.a	Server pro COBOL
libmqicb.a	Klient pro COBOL
libimqc23ia.a	Klient pro C++
libimqs23ia.a	Server pro C++

V aplikaci se závitem se připojte k jedné z následujících knihoven:

Tabulka 90. Soubory knihovny pro aplikace AIX s podporou podprocesů. Tabulka se dvěma sloupci, která vypisuje soubory knihovny a prostředí pro každý soubor knihovny.	
Soubor knihovny	**Prostředí**
libmqm_r.a	Server pro C
libmqic_r.a & **libmqm_r.a**	Klient pro C
libmqmzf_r.a	Výstupy instalovatelné služby pro C
libmqmxa_r.a	Rozhraní XA serveru
libmqmxa64_r.a	Alternativní rozhraní XA serveru
libmqcxa_r.a	Rozhraní XA klienta
libmqcxa64_r.a	Alternativní rozhraní XA klienta
libimqc23ia_r.a	Klient pro C++
libimqs23ia_r.a	Server pro C++

Poznámka: Nemůžete odkazovat na více než jednu knihovnu. To znamená, že nemůžete současně propojit jak se závity, tak i bez podprocesové knihovny.

IBM MQ for HP-UX

V systému IBM MQ for HP-UXmusíte propojit svůj program se soubory knihovny MQI dodanými pro prostředí, ve kterém spouštíte aplikaci, kromě těch, které poskytuje operační systém.

Platforma IA64 (IPF)

V aplikaci bez podprocesů se připojte k jedné z následujících knihoven:

Tabulka 91. Soubory knihovny pro aplikace bez podprocesů HP-UX	
Soubor knihovny	**Prostředí**
libmqm.so	Server pro C
libmqic.so & **libmqm.so**	Klient pro C
libmqmzf.so	Výstupy instalovatelné služby pro C
libmqmxa.so	Rozhraní XA serveru
libmqmxa64.so	Alternativní rozhraní XA serveru
libmqcxa.so	Rozhraní XA klienta
libmqcxa64.so	Alternativní rozhraní XA klienta
libimqi23ah.so	JAZYK C++
libmqmcbrt.o	Běhová knihovna produktu IBM MQ pro podporu Micro Focus COBOL
libmqmcb.so	Server pro COBOL
libmqicb.so	Klient pro COBOL

V aplikaci se závitem se připojte k jedné z následujících knihoven:

Tabulka 92. Soubory knihovny pro aplikace HP-UX s podporou podprocesů	
Soubor knihovny	**Prostředí**
libmqm_r.so	Server pro C
libmqmzf_r.so & **libmqm_r.so**	Výstupy instalovatelné služby pro C
libmqmxa_r.so	Rozhraní XA serveru
libmqmxa64_r.so	Alternativní rozhraní XA serveru
libmqcxa_r.so	Rozhraní XA klienta
libmqcxa64_r.so	Alternativní rozhraní XA klienta
libimqi23ah_r.so	JAZYK C++

Poznámka: Nemůžete odkazovat na více než jednu knihovnu. To znamená, že nemůžete současně propojit jak se závity, tak i bez podprocesové knihovny.

IBM i *IBM MQ for IBM i*

V produktu IBM MQ for IBM ipropojte svůj program se soubory knihovny MQI dodanými pro prostředí, ve kterém spouštíte svoji aplikaci, navíc k souborům poskytujícím operační systém.

Pro aplikace bez podprocesů:

Tabulka 93. Soubory knihovny pro aplikace bez podprocesů IBM i	
Soubor knihovny	**Prostředí**
LIBMQM	Servisní program serveru a klienta
LIBMQIC	Klientský servisní program
IMQB23I4	Základní servisní program C++
IMQS23I4	Servisní program serveru C++
LIBMQMZF	Instalovatelné uživatelské procedury pro C

V aplikaci se závitem:

Tabulka 94. Soubory knihovny pro aplikace IBM i s podporou podprocesů	
Soubor knihovny	**Prostředí**
LIBMQM_R	Servisní program serveru & klienta
IMQB23I4_R	Základní servisní program C++
IMQS23I4_R	Servisní program serveru C++
LIBMQMZF_R	Instalovatelné uživatelské procedury pro C
LIBMQIC_R	Klientský servisní program

V systému IBM MQ for IBM imůžete psát své aplikace v jazyce C + +. Chcete-li vidět, jak odkazovat na aplikace C + +, a chcete-li získat úplné podrobnosti o všech aspektech použití jazyků C + +, podívejte se na téma Použití C++.

Linux *IBM MQ for Linux*

V systému IBM MQ for Linuxmusíte propojit svůj program se soubory knihovny MQI dodanými pro prostředí, ve kterém spouštíte aplikaci, kromě těch, které poskytuje operační systém.

V aplikaci bez podprocesů se připojte k jedné z následujících knihoven:

Tabulka 95. Soubory knihovny pro aplikace bez podprocesů Linux	
Soubor knihovny	**Prostředí**
libmqm.so	Server pro C
libmqic.so & **libmqm.so**	Klient pro C
libmqmzf.so	Výstupy instalovatelné služby pro C
libmqmxa.so	Rozhraní XA serveru
libmqmxa64.so	Alternativní rozhraní XA serveru
libmqcxa.so	Rozhraní XA klienta
libmqcxa64.so	Alternativní rozhraní XA klienta
libimqc23gl.so	Klient pro C++
libimqs23gl.so	Server pro C++

V aplikaci se závitem se připojte k jedné z následujících knihoven:

Tabulka 96. Soubory knihovny pro aplikace Linux s podporou podprocesů	
Soubor knihovny	**Prostředí**
libmqm_r.so	Server pro C
libmqic_r.so & **libmqm_r.so**	Klient pro C
libmqmzf_r.so	Výstupy instalovatelné služby pro C
libmqmxa_r.so	Rozhraní XA serveru
libmqmxa64_r.so	Alternativní rozhraní XA serveru
libmqcxa_r.so	Rozhraní XA klienta
libmqcxa64_r.so	Alternativní rozhraní XA klienta
libimqc23gl_r.so	Klient pro C++
libimqs23gl_r.so	Server pro C++

Poznámka: Nemůžete odkazovat na více než jednu knihovnu. To znamená, že nemůžete současně propojit jak se závity, tak i bez podprocesové knihovny.

Solaris *IBM MQ for Solaris*

V systému IBM MQ for Solarismusíte propojit svůj program se soubory knihovny MQI dodanými pro prostředí, ve kterém spouštíte svoji aplikaci, navíc k souborům poskytujícím operační systém.

Tabulka 97. Soubory knihovny pro aplikace produktu Solaris	
Soubor knihovny	**Prostředí**
libmqm.so	Server a klient pro C
libmqmzse.so	Pro C
libmqic.so	Klient pro C
libmqmcs.so	Běžné služby pro jazyk C
libmqmzf.so	Výstupy instalovatelné služby pro C
libmqmxa.so	Rozhraní XA serveru

Tabulka 97. Soubory knihovny pro aplikace produktu Solaris (pokračování)	
Soubor knihovny	**Prostředí**
libmqmxa64.so	Alternativní rozhraní XA serveru
libmqcxa.so	Rozhraní XA klienta
libmqcxa64.so	Alternativní rozhraní XA klienta
libimqc23as.a	Klient pro C++
libimqs23as.a	Server pro C++

Windows *IBM MQ for Windows*

V systému IBM MQ for Windowsmusíte propojit svůj program se soubory knihovny MQI dodanými pro prostředí, ve kterém spouštíte svoji aplikaci, navíc k souborům poskytnutým operačním systémem:

Tabulka 98. Soubory knihovny pro aplikace produktu Windows	
Soubor knihovny	**Prostředí**
MQ_INSTALLATION_PATH\Tools\Lib\mqm.lib	Server for C (32bitový)
MQ_INSTALLATION_PATH\Tools\Lib\mqic.lib	Klient pro C (32bitový)
MQ_INSTALLATION_PATH\Tools\Lib\mqmxa.lib	Rozhraní XA serveru pro C (32bitový)
MQ_INSTALLATION_PATH\Tools\Lib\mqcxa.lib	Rozhraní XA klienta pro C (32bitové)
MQ_INSTALLATION_PATH\Tools\Lib\mqicxa.lib	Klient MTS pro C (32bitový)
MQ_INSTALLATION_PATH\Tools\Lib\mqmcics4.lib 32	Podpora serveru TXSeries CICS pro C (32bitová verze)
MQ_INSTALLATION_PATH\Tools\Lib\mqccics4.lib 32	Podpora klienta TXSeries CICS pro C (32bitová verze)
MQ_INSTALLATION_PATH\Tools\Lib\mqmzf.lib	Instalovatelné služby jsou ukončeny pro C (32bitový)
MQ_INSTALLATION_PATH\Tools\Lib\mqmcbb.lib	Server pro IBM COBOL (32bitový)
MQ_INSTALLATION_PATH\Tools\Lib\mqmcb.lib	Server pro Micro Focus COBOL (32bitový)
MQ_INSTALLATION_PATH\Tools\Lib\mqiccbb.lib	Klient pro IBM COBOL (32bitový)
MQ_INSTALLATION_PATH\Tools\Lib\mqiccb.lib	Klient pro Micro Focus COBOL (32bitový)
MQ_INSTALLATION_PATH\Tools\Lib\imqs23vn.lib	Server pro C++ (32bitový)
MQ_INSTALLATION_PATH\Tools\Lib\imqc23vn.lib	Klient pro C++ (32bitový)
MQ_INSTALLATION_PATH\Tools\Lib\imqb23vn.lib	Základ pro C++ (32bitový)
MQ_INSTALLATION_PATH\Tools\Lib\imqx23vn.lib	Klient MTS pro C++ (32bitový)
MQ_INSTALLATION_PATH\Tools\Lib64\mqm.lib	Server for C (64bitový)
MQ_INSTALLATION_PATH\Tools\Lib64\mqic.lib	Klient pro C (64bitový)
MQ_INSTALLATION_PATH\Tools\Lib64\mqmxa.lib	Rozhraní XA serveru pro C (64bitový)
MQ_INSTALLATION_PATH\Tools\Lib64\mqcxa.lib	Rozhraní XA klienta pro C (64 bitů)
MQ_INSTALLATION_PATH\Tools\Lib64\mqicxa.lib	Klient MTS pro C (64bitový)
MQ_INSTALLATION_PATH\Tools\Lib64\mqmcbb.lib	Server pro IBM COBOL (64bitový)

Tabulka 98. Soubory knihovny pro aplikace produktu Windows (pokračování)	
Soubor knihovny	**Prostředí**
MQ_INSTALLATION_PATH\Tools\Lib64\mqmcb.lib	Server pro Micro Focus COBOL (64bitový)
MQ_INSTALLATION_PATH\Tools\Lib64\mqiccbb.lib	Klient pro IBM COBOL (64bitový)
MQ_INSTALLATION_PATH\Tools\Lib64\mqiccb.lib	Klient pro Micro Focus COBOL (64bitový)
MQ_INSTALLATION_PATH\Tools\Lib64\imqs23vn.lib	Server pro C++ (64bitový)
MQ_INSTALLATION_PATH\Tools\Lib64\imqc23vn.lib	Klient pro C++ (64bitový)
MQ_INSTALLATION_PATH\Tools\Lib64\imqb23vn.lib	Základ pro C++ (64bitový)
MQ_INSTALLATION_PATH\Tools\Lib64\imqx23vn.lib	Klient MTS pro C++ (64bitový)

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Použijte `amqmdnet.dll` pro kompilaci .NET programů. Další informace naleznete v části "Kompilace programů produktu IBM MQ .NET" na stránce 558 v části "Vývoj aplikací produktu .NET" na stránce 514 .

Tyto soubory se dodávají kvůli kompatibilitě s předchozími vydáními:

```
mqic32.lib
mqic32xa.lib
```

z/OS *IBM MQ for z/OS programy stubu*

Než budete moci spustit program napsaný s produktem IBM MQ for z/OS, musíte jej propojit s programem typu stub, který je dodáván s produktem IBM MQ for z/OS pro prostředí, ve kterém spouštíte aplikaci.

Stub poskytuje první fázi zpracování vašich volání do požadavků, které může produkt IBM MQ for z/OS zpracovat.

Produkt IBM MQ for z/OS poskytuje následující stuby:

CSQBSTUB
Stub program pro dávkové programy z/OS

CSQBRRSI
Stub program pro dávkové programy produktu z/OS využívající službu RRS prostřednictvím rozhraní MQI

CSQBRSTB
Stub program pro dávkové programy produktu z/OS využívající službu RRS přímo

CSQCSTUB
Stub program pro programy CICS

STUB CSQQSTUB
Stub program pro programy IMS

CSQXSTUB
Stub program for distributed queuing non-CICS exits

CSQASTUB
Program stub pro ukončení převodu dat

Upozornění: Pokud použijete program stub jiný než jeden uvedený pro určité prostředí, může mít nepředvídatelné výsledky.

Poznámka: Používáte-li program stubu CSQBRSTB, proveďte linkování s ATRSCSS z SYS1.CSSLIB. (SYS1.CSSLIB je také znám jako *Knihovna Callable Services Library*). Další informace o RRS najdete v tématu "Správa transakcí a zotavitelné služby správce prostředků" na stránce 822.

Případně můžete stub dynamicky volat ze svého programu. Tato technika je popsána v tématu "Dynamické volání stubu IBM MQ" na stránce 1007.

V produktu IMSmůžete také potřebovat použít speciální modul rozhraní jazyka, který je dodáván s produktem IBM MQ.

Nespouštějte aplikace, které jsou upravovány odkazem s CSQBSTUB a CSQQSTUB ve stejné oblasti MPP IMS . To může způsobit problémy, jako například zprávy DFS3607I nebo CSQQ005E . První volání MQCONN v adresním prostoru určuje, které rozhraní bude použito, a transakce CSQQSTUB a CSQBSTUB se musí spustit v různých oblastech zpráv produktu IMS .

Parametry společné pro všechna volání

Pro všechna volání jsou společné dva typy parametrů: popisovače a návratové kódy.

Použití popisovačů

Všechna volání MQI používají jeden nebo více *manipulátorů*. Ty identifikují správce front, frontu nebo jiný objekt, zprávu nebo odběr, jak je to vhodné pro volání.

Aby program mohl komunikovat se správcem front, musí mít tento program jedinečný identifikátor, v němž zná správce front. Tento identifikátor se nazývá *manipulátor připojení*, někdy označovaný také jako *Hconn*. Pro programy CICS je manipulátor připojení vždy nula. Pro všechny ostatní platformy nebo styly programů je manipulátor připojení vrácen voláním MQCONN nebo MQCONNX při připojování programu ke správci front. Programy předávají obslužnou rutinu připojení jako vstupní parametr, používají-li jiná volání.

Aby mohl program pracovat s objektem IBM MQ , musí mít tento program jedinečný identifikátor, kterým tento objekt zná. Tento identifikátor se nazývá *popisovač objektu*, někdy označovaný také jako *Hobj*. Manipulátor je vrácen voláním MQOPEN, když program otevře objekt pro práci s ním. Programy předávají obslužnou rutinu objektu jako vstupní parametr, když používají následné volání MQPUT, MQGET, MQINQ, MQSET nebo MQCLOSE.

Podobně volání MQSUB vrací *popisovač odběru* nebo *Hsub*, který se používá k identifikaci odběru v následných voláních MQGET, MQCB nebo MQSUBRQ a určité výzvy ke zpracování vlastností zpráv používají *popisovač zprávy* nebo *Hmsg*.

Vysvětlení návratových kódů

Kód dokončení a kód příčiny jsou vráceni jako výstupní parametry každým voláním. Ty jsou souhrnně označovány jako *návratové kódy*.

Chcete-li zobrazit, zda je volání úspěšné, každé volání vrátí při dokončení volání *kód dokončení* . Kód dokončení je obvykle buď MQCC_OK označující úspěch, nebo MQCC_FAILED indikující selhání. Některá volání mohou vrátit přechodný stav, MQCC_WARNING, označující dílčí úspěch.

Každé volání také vrací *kód příčiny* , který zobrazuje příčinu selhání nebo částečného úspěchu volání. Existuje mnoho kódů příčiny, které se za takových okolností vztahují k zaplnění fronty, operace získání není pro frontu povolena a pro správce front není definována konkrétní fronta. Programy mohou použít kód příčiny k rozhodnutí, jak pokračovat. Mohou například vyzvat uživatele ke změně svých vstupních dat, pak znovu zavolat, nebo mohou vrátit chybovou zprávu uživateli.

Je-li kód dokončení MQCC_OK, kód příčiny je vždy MQRC_NONE.

Vyplnění a kódy příčiny pro každé volání jsou uvedeny s popisem tohoto volání. Viz Call descriptions a vyberte příslušné volání ze seznamu.

Podrobnější informace, včetně nápadů pro nápravnou akci, najdete v tématu:

- z/OS položky Zprávy, dokončení a kódy příčiny produktu IBM MQ for z/OS pro IBM MQ for z/OS
- Zprávy a kódy příčin pro všechny ostatní platformy IBM MQ

Určení vyrovnávacích pamětí

Správce front se odkazuje na vyrovnávací paměti pouze v případě, že jsou vyžadovány. Pokud nevyžadujete vyrovnávací paměť při volání nebo vyrovnávací paměť má nulovou délku, můžete použít ukazatel null na vyrovnávací paměť.

Při zadávání velikosti vyrovnávací paměti, kterou vyžadujete, vždy použijte délku dat.

Použijete-li vyrovnávací paměť k uložení výstupu z volání (například k zadržení dat zprávy pro volání MQGET nebo hodnoty atributů dotazovaných voláním MQINQ), správce front se pokusí o vrácení kódu příčiny, pokud zadaná vyrovnávací paměť není platná nebo je v úložišti jen pro čtení. Avšak nemusí být vždy schopen vrátit kód příčiny.

z/OS *Aspekty dávky produktu z/OS*

z/OS dávkových programů, které volají rozhraní MQI, může být buď ve stavu supervisor, nebo ve stavu problému.

Musí však splňovat tyto podmínky:

- Musí být v režimu úlohy, nikoli v režimu SRB (service request block).
- Musí být v režimu ASC (Primary Address Space Control) (ne Access Register ASC mode).
- Nesmí se nacházet v režimu křížové paměti. Primární číslo adresního prostoru (ASN) musí být rovno sekundárnímu ASN a domovskému ASN.
- Nesmějí být používány jako výstupní programy MPF.
- Nelze zadržet žádné zámky z/OS .
- Na zásobníku FRR nemohou být žádné rutiny pro obnovu funkcí.
- Pro volání MQCONN nebo MQCONNX může být pro volání MQCONN nebo MQCONNX zadána hodnota libovolného stavového slova programu (PSW) (za předpokladu, že klíč je kompatibilní s použitím úložiště, které je v klíči TCB), ale následující volání, která používají obslužnou rutinu připojení vrácenou rozhraním MQCONN nebo MQCONNX:
  - Musí mít stejný klíč PSW, který byl použit v rámci volání MQCONN nebo MQCONNX
  - Musí mít přístupné parametry (pro zápis, je-li to vhodné) pod stejným klíčem PSW
  - Musí být vydáno v rámci stejné úlohy (TCB), ale ne v žádné podúloze úlohy
- Mohou být ve 24bitovém nebo 31bitovém adresovém režimu. Je-li však v platnosti 24bitový adresovací režim, adresy parametrů musí být interpretovány jako platné 31bitové adresy.

Není-li některá z těchto podmínek splněna, může se vyskytnout kontrola programu. V některých případech se volání nezdaří a bude vrácen kód příčiny.

Linux UNIX *Aspekty produktu UNIX and Linux*

Pokyny, které byste měli znát.

Při vývoji aplikací produktu UNIX and Linux vezměte na vědomí následující body.

Linux UNIX *Systémové volání fork v systémech UNIX and Linux*

Všimněte si těchto pokynů při použití systémového volání fork v aplikacích produktu IBM MQ .

Chce-li vaše aplikace používat produkt `fork`, nadřízený proces této aplikace by měl volat `fork` před provedením jakýchkoli volání IBM MQ , například `MQCONN`, nebo vytvořením objektu IBM MQ pomocí **`ImqQueueManager`**.

Pokud chce vaše aplikace po provedení libovolných volání IBM MQ vytvořit podřízený proces, musí kód aplikace používat `fork()` s produktem `exec()` , aby bylo zajištěno, že je podřízeným prvkem nová instance a ne přesná kopie nadřízené položky.

Pokud vaše aplikace nevyužívá produkt `exec()`, volání rozhraní API produktu IBM MQ provedené v podřízeném procesu vrátí funkci MQRC_ENVIRONMENT_ERROR.

Linux UNIX *Zpracování signálů UNIX and Linux*

Tento parametr se nevztahuje na IBM MQ for z/OS nebo IBM MQ for Windows.

Obecně se systémy UNIX, Linux a IBM i přesunuly z prostředí bez podprocesů (procesů) do vícevláknového prostředí. V prostředí bez podprocesů mohou být některé funkce implementovány pouze pomocí signálů, ačkoli většina aplikací nemusí být informována o signálech a zpracování signálů. Ve vícevláknovém prostředí podporují vlákna založená na vláknech některé z funkcí, které se používají k implementaci v prostředí bez vláken pomocí signálů.

V mnoha případech se signály a zpracování signálů, i když jsou podporovány, nehodí do vícevláknového prostředí a existují různá omezení. To může být problematické, když integrujete kód aplikace s různými knihovnami middlewaru (spuštěnými jako součást aplikace) v prostředí s podporou podprocesů, kde se každý snaží ošetřit signály. Tradiční přístup k ukládání a obnově obslužných rutin signálů (definovaných pro každý proces), který fungoval, když v rámci procesu existoval pouze jeden podproces, nefunguje v prostředí s více podprocesy. Důvodem je to, že mnoho podprocesů provedení se může pokusit o uložení a obnovení prostředku v rámci celého procesu s nepředvídatelnými výsledky.

UNIX *Aplikace bez podprocesů*

Nepoužije se na Solaris , protože všechny aplikace jsou považovány za závitové, i když používají pouze jeden podproces.

Každá funkce MQI nastavuje svůj vlastní popisovač signálu pro signály:

SIGNALRM
SIGBUS
SIGFPE
SIGSEGV
SIGILL

Popisovače uživatelů pro tyto úlohy budou nahrazeny po dobu trvání volání funkce MQI. Další signály mohou být zachyceny běžným způsobem uživatelem napsanými obslužnými rutinami. Pokud obslužnou rutinu nenainstalujete, budou na místě ponechány výchozí akce (například ignorování, výpis jádra nebo ukončení).

Poté, co IBM MQ zpracuje synchronní signál (SIGSEGV, SIGBUS, SIGFPE, SIGILL), pokusí se předat signál do libovolného registrovaného popisovače signálu před tím, než bude volání funkce MQI volat.

Linux UNIX *Aplikace s podprocesy*

Podproces je považován za připojený k produktu IBM MQ z MQCONN (nebo MQCONNX) do MQDISC.

Synchronní signály

Synchronní signály vznikají ve specifickém podprocesu.

Systémy UNIX and Linux bezpečně umožňují nastavení ovladače signálu pro takové signály pro celý proces. Produkt IBM MQ však nastaví vlastní obslužnou rutinu pro následující signály, v aplikačním procesu, zatímco jakýkoliv podproces je připojen k serveru IBM MQ:

SIGBUS
SIGFPE
SIGSEGV
SIGILL

Pokud zapisujete vícevláknové aplikace, existuje pro každý signál pouze jedna obslužná rutina signálu celého procesu. Když IBM MQ nastaví své vlastní popisovače synchronních signálů, uloží všechny dříve registrované obslužné rutiny pro každý signál. Poté, co IBM MQ zpracuje jeden z uvedených signálů, IBM MQ se pokusí zavolat popisovači signálu, který byl v platnosti v době prvního připojení IBM MQ v rámci procesu. Dříve registrované obslužné rutiny se obnoví, když se všechny podprocesy aplikace odpojí od IBM MQ.

Vzhledem k tomu, že obslužné rutiny signálů jsou ukládány a obnovovány produktem IBM MQ, aplikační podprocesy nesmí vytvářet obslužné rutiny signálů pro tyto signály, pokud existuje možnost, že je k produktu IBM MQpřipojen i jiný podproces stejného procesu.

Poznámka: Když aplikace nebo knihovna middlewaru (běžící jako část aplikace) zřídí obslužnou rutinu signálu, když je podproces připojen k produktu IBM MQ, musí obslužná rutina signálu aplikace během zpracování tohoto signálu zavolat odpovídající obslužnou rutinu IBM MQ .

Při vytváření a obnově obslužných rutin signálu musí být hlavní zásadou, že poslední manipulační program signálu, který má být uložen, musí být první, který se má obnovit:

- Když aplikace zavede manipulační program signálu po připojení k produktu IBM MQ, musí být předchozí popisovač signálu obnoven dříve, než se aplikace odpojí od IBM MQ.
- Když aplikace zavede manipulační program signálu před připojením k produktu IBM MQ, musí se před obnovením obslužné rutiny signálu odpojit od produktu IBM MQ .

Poznámka: Selhání při sledování obecné zásady, že poslední obslužná rutina signálu, která má být uložena, musí být první, která má být obnovena, může vést k neočekávanému zpracování signálů v aplikaci a potenciálně i ztrátě signálů aplikací.

Asynchronní signály

IBM MQ nepoužívá žádné asynchronní signály v vláknových aplikacích, pokud se nejedná o klientské aplikace.

Další pokyny pro klientské aplikace s podporou podprocesů

Produkt IBM MQ zpracovává následující signály během I/O na server. Tyto signály jsou definovány v komunikačním zásobníku. Aplikace nesmí vytvořit obslužnou rutinu signálů pro tyto signály, zatímco je podproces připojen ke správci front:

SIGPIPE (pro TCP/IP)

UNIX *Další pokyny při použití zpracování signálů produktu UNIX v rozhraní MQI*
Všimněte si těchto pokynů při použití zpracování signálů produktu UNIX .

Aplikace Fastpath (důvěryhodné)

Aplikace Fastpath se spouštějí ve stejném procesu jako produkt IBM MQ , a tak jsou spuštěny v prostředí s podporou podprocesů.

V tomto prostředí IBM MQ pracuje se synchronními signály SIGSEGV, SIGBUS, SIGFPE a SIGILL. Všechny ostatní signály nesmí být doručeny do aplikace Fastpath, když je připojena k produktu IBM MQ. Místo toho musí být blokovány nebo zpracovány aplikací. Pokud aplikace Fastpath zadrží takovou událost, musí být správce front zastaven a restartován, nebo může být ponechán v nedefinovaném stavu. Úplný seznam omezení pro aplikace Fastpath pod MQCONNX viz “Připojení ke správci front pomocí volání MQCONNX” na stránce 707.

Volání funkcí MQI v obslužných rutinách signálů

Když jste v popisovači signálu, nevolejte funkci MQI.

Pokusíte-li se volat funkci MQI z obslužné rutiny signálu při zpracování jiné funkce MQI, vrátí se hodnota MQRC_CALL_IN_PROGRESS. Pokud se pokusíte volat funkci MQI z obslužné rutiny signálu a přitom není aktivní žádná jiná funkce MQI, bude pravděpodobně někdy během operace selhat kvůli omezením operačního systému, kdy lze z obslužné rutiny vydat pouze výběrová volání.

Pro metody destruktoru C + +, které mohou být volány automaticky během ukončení programu, možná nebudete moci zastavit volání funkcí MQI. Ignorujte všechny chyby týkající se MQRC_CALL_IN_PROGRESS. Pokud volá obslužný program signálu exit (), příkaz IBM MQ zazálohuje nepotvrzené zprávy v bodě synchronizace jako obvykle a zavře všechny otevřené fronty.

Signály během volání MQI

Funkce MQI nevracejí kód EINTR ani žádný ekvivalent aplikačních programů.

Pokud během volání MQI dojde k signálu a volání obslužné rutiny volá *return*, volání pokračuje v tom, jako by se signál nestal. Příkaz MQGET nemůže být zejména přerušen signálem k okamžitému vrácení řízení do aplikace. Chcete-li přerušit MQGET, nastavte frontu na volbu GET_DISABLED; alternativně použijte smyčku kolem volání MQGET s konečným časovým intervalem (MQGMO_WAIT s parametrem gmo.WaitInterval) a pomocí obslužné rutiny signálu (v prostředí bez podprocesů) nebo ekvivalentní funkce v prostředí s podprocesy nastavte příznak, který přeruší smyčku.

AIX V prostředí AIX vyžaduje obslužný program IBM MQ , aby byla restartována volání systému přerušená signály. Při vytváření vašeho vlastního popisovače signálu pomocí funkce sigaction (2) nastavte příznak SA_RESTART v poli sa_flags nové struktury akce, jinak IBM MQ nemusí být schopen dokončit jakékoli volání přerušené signálem.

Uživatelské procedury a instalovatelné služby

Uživatelské procedury a instalovatelné služby, které se spouštějí jako součást procesu produktu IBM MQ v prostředí s podporou podprocesů, mají stejná omezení jako v případě aplikací se zkrácenou cestou. Považujte je za trvale připojené k produktu IBM MQ a nepoužívejte tedy signály operačního systému nebo volání mimo zajištění neporušenosti vláken.

Obslužné rutiny výstupních bodů

Uživatelé mohou instalovat obslužné rutiny ukončení pro aplikaci IBM MQ pomocí systémové služby **SYS$DCLEXH** .

Obslužná rutina ukončení obdrží řízení, když se obrázek ukončí. K ukončení obrazu se obvykle dojde, když voláte službu Konec ($EXIT) nebo Vynutit ukončení ($FORCEX). $FORCEX přeruší cílový proces v uživatelském režimu. Poté všechny obslužné rutiny uživatelských procedur v uživatelském režimu (zavedené $DCLEXH) se začnou provádět v opačném pořadí, než je zařízení. Další informace o ovladačích ukončení a $FORCEX najdete v příručce *VMS Programming Concepts Manual* a *VMS System Services Manual*.

Vočíte-li funkci MQI z obslužné rutiny ukončení, chování funkce závisí na způsobu, jakým byl obraz ukončen. Pokud byl obraz ukončen, zatímco je aktivní jiná funkce MQI, je vrácena hodnota `MQRC_CALL_IN_PROGRESS` .

Je možné volat funkci MQI z obslužné rutiny ukončení, pokud není aktivní žádná jiná funkce MQI a jsou zakázána volání upcall pro aplikaci produktu IBM MQ . Pokud jsou povolena pro aplikaci produktu IBM MQ volání upcalls, dojde k selhání s kódem příčiny `MQRC_HCONN_ERROR`.

K rozsahu volání MQCONN nebo MQCONNX se obvykle používá podproces, který jej vydal. Pokud jsou povolena volání upcalls, obslužná rutina ukončení se spustí jako oddělený podproces a popisovače připojení nemohou být sdíleny.

Obslužné rutiny ukončení jsou spuštěny v rámci přerušeného kontextu cílového procesu. Je na aplikaci, aby bylo zajištěno, že akce provedené obslužnou rutinou jsou bezpečné a spolehlivé, pro asynchronně přerušovaný kontext, ze kterého jsou volány.

Připojování k správci front a odpojování od něj

Chcete-li používat programovací služby IBM MQ , musí mít program připojení ke správci front. V této části se dozvíte, jak se připojit k správci front a odpojit je od správce front.

Způsob, jakým toto připojení závisí, závisí na platformě a prostředí, ve kterém tento program funguje:

Multi **IBM MQ for Multiplatforms**

Programy, které jsou spuštěny v těchto prostředích, mohou používat volání MQCONN MQI k připojení a volání MQDISC, které se má odpojit od správce front. Alternativně mohou programy používat volání MQCONNX.

z/OS IBM MQ for z/OS dávkové

Programy, které se spouštějí v tomto prostředí, mohou používat volání MQCONN MQI k připojení a volání MQDISC, které se má odpojit od správce front. Alternativně mohou programy používat volání MQCONNX.

z/OS dávkové programy se mohou připojovat, postupně nebo souběžně, na více správců front na stejné TCB.

z/OS IMS

Řídicí oblast IMS je při spuštění připojena k jednomu nebo více správcům front. Toto připojení je řízeno příkazy IMS . Informace o tom, jak řídit adaptér IMS v systému z/OS, viz Administrace produktu IBM MQ for z/OS. Nicméně autoři programů pro řazení zpráv do front IMS musí používat volání MQCONN MQI k určení správce front, ke kterému se mají připojit. K odpojení od tohoto správce front mohou použít volání MQDISC.

Po volání funkce IMS , která zřídí synchronizační bod, a před zpracováním zprávy pro jiného uživatele adaptér IMS zajistí, aby aplikace zavřela a odpojuje od správce front. Viz “Synchronizační body v aplikacích produktu IMS” na stránce 821.

Programy produktu IMS se mohou vzájemně nebo souběžně připojovat k více správcům front ve stejné TCB.

z/OS CICS Transaction Server pro z/OS

Programy produktu CICS nemusí provádět žádnou práci pro připojení ke správci front, protože je připojen samotný systém CICS . Toto připojení se obvykle provádí automaticky při inicializaci, ale můžete také použít transakci CKQC, která je dodána spolu s produktem IBM MQ for z/OS. Další informace o CKQC viz Administrace produktu IBM MQ for z/OS.

Úlohy produktu CICS se mohou připojit pouze ke správci front, ke kterému je oblast CICS připojena.

Programy produktu CICS mohou také používat volání MQI Connect a disconnect (MQCONN a MQDISC). Můžete to chtít udělat, abyste mohli tyto aplikace portovat do jiných prostředí nežCICS s minimem změn. Tato volání však *vždy* budou úspěšně dokončena v prostředí produktu CICS . To znamená, že návratový kód nemusí odrážet skutečný stav připojení ke správci front.

TXSeries pro Windows a otevřené systémy

Tyto programy nemusí provádět žádnou práci pro připojení ke správci front, protože je připojen samotný systém CICS . Proto je podporováno pouze jedno připojení v daném okamžiku. Aplikace produktu CICS musí při volání funkce MQCONN vyvolat volání MQCONN a volání MQDISC před jejich ukončením.

Chcete-li zjistit více o připojení a odpojení od správce front, použijte následující odkazy:

- “Připojení ke správci front pomocí volání MQCONN” na stránce 706
- “Připojení ke správci front pomocí volání MQCONNX” na stránce 707
- “Odpojení programů od správce front pomocí příkazu MQDISC” na stránce 712

Související pojmy

“Přehled rozhraní fronty zpráv” na stránce 690
Zde jsou uvedeny informace o komponentách rozhraní MQI (Message Queue Interface).

“Otevírání a zavírání objektů” na stránce 712
Tyto informace poskytují vhled do otevírání a zavírání objektů IBM MQ .

“Vložení zpráv do fronty” na stránce 723
V této části se dozvíte, jak vložit zprávy do fronty.

“Získávání zpráv z fronty” na stránce 737
Tyto informace použijte k získání informací o získávání zpráv z fronty.

“Inquaning about a nastavení atributů objektu” na stránce 815
Atributy jsou vlastnosti, které definují charakteristiky objektu IBM MQ .

“Potvrzení a zálohování jednotek práce” na stránce 818

Tyto informace popisují, jak potvrdit a zazálohovat všechny zotavitelné operace get a put, které se vyskytly v jednotce práce.

“Spuštění aplikací produktu IBM MQ pomocí spouštěčů” na stránce 829
Informace o spouštěčích a o tom, jak spustit aplikace IBM MQ pomocí spouštěčů.

“Práce s MQI a klastry” na stránce 847
Existují speciální volby na voláních a návratových kódech, které se vztahují ke klastrování.

“Použití a zápis aplikací v systému IBM MQ for z/OS” na stránce 851
Aplikace produktu IBM MQ for z/OS mohou být vytvořeny z programů spuštěných v mnoha různých prostředích. To znamená, že mohou využít výhody zařízení dostupných ve více než jednom prostředí.

“Aplikace mostu IMS a IMS v systému IBM MQ for z/OS” na stránce 57
Tyto informace vám pomohou při psaní aplikací produktu IMS pomocí produktu IBM MQ.

Připojení ke správci front pomocí volání MQCONN

Pomocí těchto informací se naučíte, jak se připojit ke správci front pomocí volání MQCONN.

Obecně lze připojit buď ke specifickému správci front, nebo k výchozímu správci front:

- Pro databázi IBM MQ for z/OSje v dávkovém prostředí určen výchozí správce front v modulu CSQBDEFV.
- Pro systémy IBM MQ for Windows, IBM i, UNIXa Linux je výchozí správce front určen v souboru mqs.ini .

Eventuálně v prostředí z/OS MVS dávek, TSO a RRS se můžete připojit k libovolnému správci front v rámci skupiny sdílení front. Požadavek MQCONN nebo MQCONNX vybere jednoho z aktivních členů skupiny.

Když se připojíte ke správci front, musí být lokální pro danou úlohu. Musí náležet ke stejnému systému jako aplikace produktu IBM MQ .

V prostředí IMS musí být správce front připojen k řídicí oblasti IMS a k závislé oblasti, kterou tento program používá. Výchozí správce front je určen v modulu CSQQDEFV, je-li nainstalován produkt IBM MQ for z/OS .

S prostředím TXSeries CICS a TXSeries for Windows a AIXmusí být správce front definován jako prostředek XA pro CICS.

Chcete-li se připojit k výchozímu správci front, volejte MQCONN a uveďte název sestávající zcela z mezer nebo začněte znakem null (X'00 ').

Aplikace musí být autorizována, aby se mohla úspěšně připojit ke správci front. Další informace viz Zabezpečení.

Výstup z MQCONN je:

- Popisovač připojení (**Hconn**).
- Kód dokončení
- Kód příčiny

Použít obslužnou rutinu připojení při následných voláních MQI.

Pokud kód příčiny informuje o tom, že aplikace je již připojena k tomuto správci front, je vrácený popisovač připojení stejný jako ten, který byl vrácen při prvním připojení aplikace. Aplikace nesmí v této situaci vyvolat volání MQDISC, protože volající aplikace očekává, že zůstane připojena.

Rozsah manipulátoru připojení je stejný jako obor manipulátoru objektu (viz “Otevírání objektů pomocí volání MQOPEN” na stránce 714).

Popisy parametrů jsou uvedeny v popisu volání MQCONN v MQCONN.

Volání MQCONN selže, pokud je správce front při zadávání volání do klidového stavu nebo pokud se správce front vypíná.

Rozsah MQCONN nebo MQCONNX

K rozsahu volání MQCONN nebo MQCONNX se obvykle používá podproces, který jej vydal. To znamená, že manipulátor připojení vrácený z volání je platný pouze v rámci podprocesu, který vydal volání. V jednom okamžiku může být pomocí manipulátoru provedeno pouze jedno volání. Je-li použit z jiného podprocesu, je odmítnut jako neplatný. Máte-li ve své aplikaci více podprocesů a každý chce používat volání IBM MQ , musí každý z nich zadat volání MQCONN nebo MQCONNX.

Pokud proces provádí více volání MQCONN, není třeba pro každé volání, které má být provedeno do stejného správce front. V daném okamžiku však lze z podprocesu vytvořit pouze jedno připojení produktu IBM MQ . Případně zvažte možnost “Sdílená připojení (nezávislá na podprocesech) s MQCONNX” na stránce 710 , chcete-li povolit více připojení produktu IBM MQ z jednoho podprocesu a připojení IBM MQ pro použití z libovolného podprocesu.[8]

Je-li vaše aplikace spuštěna jako klient, může se připojit k více než jednomu správci front v rámci podprocesu.

Připojení ke správci front pomocí volání MQCONNX

Volání MQCONNX je podobné volání MQCONN, ale obsahuje volby pro řízení toho, jak volání funguje.

Jako vstup pro objekt MQCONNX můžete zadat název správce front z/OS nebo název skupiny sdílení front v systémech sdílených front produktu z/OS.

Výstup z MQCONNX je:

- Popisovač připojení (Hconn)
- Kód dokončení
- Kód příčiny

Manipulátor připojení se používá při následných voláních MQI.

Popis všech parametrů MQCONNX je uveden v MQCONNX. Pole *Options* umožňuje nastavit parametry STANDARD_BINDING, FASTPATH_BINDING, SHARED_BINDING nebo ISOLATED_BINDING pro libovolnou verzi MQCNO. Pomocí volání MQCONNX můžete také vytvořit sdílená připojení (nezávislé na podprocesech). Další informace o těchto tématech viz “Sdílená připojení (nezávislá na podprocesech) s MQCONNX” na stránce 710 .

VAZBA MQCNO_STANDARD_BINDING

MQCONNX (například MQCONN) je standardně tvořen dvěma logickými podprocesy, ve kterých se aplikace IBM MQ a lokální agent správce front spouštějí v samostatných procesech. Aplikace IBM MQ požádá o operaci IBM MQ a lokální správce front, který obsluhuje požadavek. Toto je definováno volbou MQCNO_STANDARD_BINDING na volání MQCONNX.

Zadáte-li MQCNO_STANDARD_BINDING, volání MQCONNX používá buď MQCNO_SHARED_BINDING, nebo MQCNO_ISOLATED_BINDING, v závislosti na hodnotě atributu **DefaultBindType** správce front, který je definován v souboru qm.ini.

Toto je výchozí hodnota.

Pokud odkazujete na knihovnu produktu `mqm` , je nejprve proveden pokus o použití standardního připojení k serveru s použitím výchozího typu vazby. Pokud se základní knihovna serveru nepodařilo načíst, bude namísto toho proveden pokus o připojení klienta.

- Je-li zadána proměnná prostředí MQ_CONNECT_TYPE, může být dodána jedna z následujících voleb pro změnu chování MQCONN nebo MQCONNX, je-li zadáno MQCNO_STANDARD_BINDING. (Výjimkou je, je-li hodnota MQCNO_FASTPATH_BINDING zadána s hodnotou MQ_CONNECT_TYPE

[8] Při použití aplikací s podporou podprocesů s produktem IBM MQ v systémech UNIX and Linux je třeba zajistit, aby aplikace měly dostatečnou velikost zásobníku pro podprocesy. Zvažte použití velikosti zásobníku 256 KB nebo větší, když aplikace s podporou podprocesů provádějí volání MQI buď samostatně, nebo s jinými obslužnými rutinami signálu (například CICS).

nastaveným na hodnotu LOCAL nebo STANDARD , aby umožnila snížení úrovně zkrácené cesty administrátorem bez související změny aplikace:

Hodnota	Význam
CLIENT	Pokus o připojení klienta se provede pouze.
Rychlý	Tato hodnota byla v předchozích verzích podporována, ale je nyní ignorována, pokud byla zadána.
LOKÁLNÍ	Pokus o připojení k serveru se pouze pokusil. Připojení zrychleného přístupu se sníží na standardní připojení k serveru.
STANDARD	Podporováno z důvodu kompatibility s předchozími verzemi. Tato hodnota je nyní považována za LOCAL.

- Není-li proměnná prostředí MQ_CONNECT_TYPE nastavena při volání MQCONN, je proveden pokus o připojení standardního serveru s použitím výchozího typu vazby. Pokud se knihovna serveru nepodaří načíst, dojde k pokusu o připojení klienta.

VAZBA MQCNO_FASTPATH_BINDING

Důvěryhodné aplikace znamenají, že se aplikace IBM MQ a lokální agent správce front stanou stejným procesem. Vzhledem k tomu, že proces agenta již nemusí pro přístup ke správci front používat rozhraní, tyto aplikace se stanou rozšířením správce front. Toto je definováno volbou MQCNO_FASTPATH_BINDING v rámci volání MQCONNX.

Musíte propojit důvěryhodné aplikace se závitem IBM MQ knihoven. Pokyny, jak nastavit aplikaci produktu IBM MQ tak, aby se spouštělo jako důvěryhodné, naleznete v tématu Volby MQCNO.

Tato volba poskytuje nejvyšší výkon.

Poznámka: Tato volba ohrožuje integritu správce front: neexistuje žádná ochrana proti přepsání jeho paměti. To platí také v případě, že aplikace obsahuje chyby, které mohou být vystaveny i zprávám a dalším datům ve správci front. Před použitím této volby zvažte tyto problémy.

CQCNO_SHARED_BINDING

Tuto volbu určete, chcete-li aplikaci a lokálního agenta správce front spouštět v samostatných procesech. Tím je zachována integrita správce front, tj. chrání správce front před chybnými programy. Avšak aplikace a lokální agent správce front sdílejí některé prostředky.

Tato volba je mezilehlá mezi MQCNO_FASTPATH_BINDING a MQCNO_ISOLATED_BINDING, a to jak z hlediska ochrany integrity správce front, tak i z hlediska výkonu volání MQI.

Objekt MQCNO_SHARED_BINDING je ignorován, pokud správce front nepodporuje tento typ vazby. Zpracování pokračuje, jako by tato volba nebyla uvedena.

Pokud se aplikace připojila k lokálnímu správci front pomocí MQCNO_SHARED_BINDING, může být správce front zastaven za běhu aplikace. Pokud správce front restartujete v době, kdy je aplikace stále spuštěna, dojde k selhání pokusu o spuštění správce front s chybou AMQ7018 , protože aplikace stále zadržuje prostředky, které potřebuje správce front.

Chcete-li spustit správce front, je třeba aplikaci ukončit.

VAZBA MQCNO_ISOLATED_BINDING

Zadáním této volby provedete běh aplikace a lokálního agenta správce front v oddělených procesech, jako například pro MQCNO_SHARED_BINDING. V tomto případě je však aplikační proces a lokální agent správce front izolováni od sebe navzájem, protože nesdílejí prostředky.

Toto je nejbezpečnější volba pro ochranu integrity správce front, ale poskytuje nejpomalejší výkon volání MQI.

Objekt MQCNO_ISOLATED_BINDING je ignorován, pokud správce front nepodporuje tento typ vazby. Zpracování pokračuje, jako by tato volba nebyla uvedena.

VÁZÁNÍ MQCNO_CLIENT_BINDING

Tuto volbu uveďte, chcete-li aplikaci pokusit pouze o připojení klienta. Tato volba má následující omezení:

- z/OS Parametr MQCNO_CLIENT_BINDING je v produktu z/OSignorován.
- Hodnota MQCNO_CLIENT_BINDING byla odmítnuta s chybou MQRC_OPTIONS_ERROR, je-li zadána s jinou volbou vazby MQCNO, než je MQCNO_STANDARD_BINDING.
- MQCNO_CLIENT_BINDING není k dispozici pro produkt Java , protože má vlastní mechanizmy pro výběr typu vazby.
- Není-li proměnná prostředí MQ_CONNECT_TYPE nastavena při volání MQCONN, je proveden pokus o připojení standardního serveru s použitím výchozího typu vazby. Pokud se knihovna serveru nepodaří načíst, dojde k pokusu o připojení klienta.

MQCNO_LOCAL_BINDING.

Tuto volbu uveďte, chcete-li provést pokus aplikace o připojení k serveru. Je-li také zadán buď MQCNO_FASTPATH_BINDING, MQCNO_ISOLATED_BINDING nebo MQCNO_SHARED_BINDING, bude místo toho připojení tohoto typu a v této sekci je zdokumentováno. Jinak se provede pokus o standardní připojení k serveru s použitím výchozího typu vazby. Objekt MQCNO_LOCAL_BINDING má následující omezení:

- z/OS Parametr MQCNO_LOCAL_BINDING je v produktu z/OSignorován.
- MQCNO_LOCAL_BINDING byl odmítnut s chybou MQRC_OPTIONS_ERROR, pokud je zadán spolu s jinou volbou MQCNO reconnect jiným než MQCNO_RECONNECT_AS_DEF.
- MQCNO_LOCAL_BINDING není k dispozici pro produkt Java , protože má vlastní mechanizmy pro výběr typu vazby.
- Není-li proměnná prostředí MQ_CONNECT_TYPE nastavena při volání MQCONN, je proveden pokus o připojení standardního serveru s použitím výchozího typu vazby. Pokud se knihovna serveru nepodaří načíst, dojde k pokusu o připojení klienta.

z/OS Na z/OS jsou tyto volby tolerovány, ale provádí se pouze standardní vázané spojení.

z/OS MQCNO verze 3, pro produkt z/OS, povoluje čtyři různé volby:

MQCNO_SERIALIZE_CONN_TAG_QSG

To umožňuje aplikaci požadovat, aby v jedné skupině sdílení front byla spuštěna pouze jedna instance aplikace v jednom okamžiku. Toho lze dosáhnout registrací použití značky připojení s hodnotou, která je určena nebo odvozena z aplikace. Značka je 128bajtový znakový řetězec určený ve verzi 3 MQCNO.

MQCNO_RESTRICT_CONN_TAG_QSG

Tato hodnota se používá v případech, kdy se aplikace skládá z více než jednoho procesu (nebo z TCB), z nichž každý se může připojit ke správci front. Připojení je povoleno pouze v případě, že neexistuje žádné aktuální použití značky, nebo žádost o aplikaci se nachází ve stejném rozsahu zpracování. Jedná se o adresní prostor MVS v rámci stejné skupiny sdílení front jako vlastník značky.

MQCNO_SERIALIZE_CONN_TAG_Q_MGR

Je to podobné jako MQCNO_SERIALIZE_CONN_TAG_QSG, ale pouze lokální správce front je dotazován, aby zjistil, zda již požadovaná značka již je používána.

MQCNO_RESTRICT_CONN_TAG_Q_MGR

Tento postup je podobný objektu MQCNO_RESTRICT_CONN_TAG_QSG, ale pouze lokální správce front je dotazován, aby bylo možné zjistit, zda již požadovaná značka již je používána.

Omezení pro důvěryhodné aplikace

Pro důvěryhodné aplikace platí následující omezení:

- Důvěryhodné aplikace je třeba explicitně odpojit od správce front.
- Před ukončením správce front s příkazem endmqm musíte zastavit důvěryhodné aplikace.
- Musíte použít asynchronní signály a přerušení časovače (například `sigkill`) s MQCNO_FASTPATH_BINDING.
- Na všech platformách se podproces v rámci důvěryhodné aplikace nemůže připojit ke správci front, zatímco jiný podproces ve stejném procesu je připojen k jinému správci front.
- V systému IBM MQ v systémech UNIX and Linux je třeba pro všechna volání MQI použít skupinu mqm jako platné hodnoty userID a groupID (všechny volání MQI). Tato ID můžete změnit před provedením volání jiného než MQI vyžadujícího ověření (například při otevření souboru), ale před provedením dalšího volání MQI jej musíte změnit zpět na skupinu mqm.
- IBM i V systému IBM MQ for IBM i:
  1. Důvěryhodné aplikace musí být spuštěny pod uživatelským profilem QMQM. Není dostatečné, aby uživatelský profil byl členem skupiny QMQM nebo že tento program adoptuje oprávnění QMQM. Možná nebude možné použít uživatelský profil QMQM k přihlášení k interaktivním úlohám nebo k uvedení v popisu úlohy pro úlohy spuštěné důvěryhodnými aplikacemi. V tomto případě jeden přístup používá funkce odkládání rozhraní API produktu IBM i , QSYGETPH, QWTSETP a QSYRLSPH pro dočasnou změnu aktuálního uživatele úlohy na QMQM, zatímco programy MQ běží. Podrobnosti o těchto funkcích spolu s příkladem jejich použití jsou uvedeny v části Security API v příručce *IBM i System API Reference*.
  2. Nepokoušejte se zrušit důvěryhodné aplikace pomocí volby 2 systémového požadavku nebo ukončením úloh, ve kterých jsou spuštěny pomocí příkazu ENDJOB.
- U IBM MQ for HP-UXje pravděpodobné, že aplikace s rychlou podporou podprocesů budou muset nastavit větší velikost zásobníku, než je výchozí. Použijte velikost 256 kB.
- V IBM MQ for Windows důvěryhodných 64bitových aplikacích nejsou podporovány. Pokusíte-li se spustit důvěryhodnou 64-bitovou aplikaci, bude snížena úroveň na standardní vázané připojení.
- V systémech IBM MQ na systémech UNIX and Linux se nedůvěryhodné 32bitové aplikace nepodporují. Pokusíte-li se spustit důvěryhodnou 32bitovou aplikaci, bude snížena úroveň na standardní vázané připojení.

Sdílená připojení (nezávislá na podprocesech) s MQCONNX
Tyto informace použijte k seznámení se se sdílenými připojeními s rozhraním MQCONNX a s některými poznámkami k použití, které je třeba zvážit.

Poznámka: Nepodporováno na IBM MQ for z/OS.

Na jiných platformách produktu IBM MQ než IBM MQ for z/OSje připojení, které bylo vytvořeno s produktem `MQCONN` , dostupné pouze pro podproces, který vytvořil připojení. Volby v rámci volání MQCONNX umožňují vytvořit připojení, které může být sdíleno všemi podprocesy v procesu. Je-li aplikace spuštěna v transakčním prostředí, které vyžaduje zadání volání MQI ve stejném podprocesu, je třeba použít následující výchozí volbu:

MQCNO_HANDLE_SHARE_NONE
Vytvoří nesdílené připojení.

Ve většině ostatních prostředí můžete použít jednu z následujících možností sdíleného připojení podprocesů:

MQCNO_HANDLE_SHARE_BLOCK
Vytvoří sdílené připojení. Je-li v připojení k produktu `MQCNO_HANDLE_SHARE_BLOCK` aktuálně používáno připojení prostřednictvím volání MQI v jiném podprocesu, bude volání MQI čekat, dokud nebude dokončeno aktuální volání MQI.

MQCNO_HANDLE_SHARE_NO_BLOCK
Vytvoří sdílené připojení. Je-li v připojení k produktu MQCNO_HANDLE_SHARE_NO_BLOCK připojení aktuálně používáno voláním MQI v jiném podprocesu, volání MQI se okamžitě nezdaří s příčinou MQRC_CALL_IN_PROGRESS.

S výjimkou prostředí MTS (Microsoft Transaction Server) je výchozí hodnota MQCNO_HANDLE_SHARE_NONE. V prostředí MTS je předvolená hodnota MQCNO_HANDLE_SHARE_BLOCK.

Popisovač připojení je vrácen z volání MQCONNX . Popisovač může být použit následujícími voláními MQI z libovolného podprocesu v procesu, přidružování těchto volání k obslužné rutiny vrácené z produktu MQCONNX. Volání MQI s použitím jednoho sdíleného manipulátoru jsou serializována mezi podprocesy.

Například následující posloupnost aktivit je možná se sdíleným hantem:

1. Vlákno 1 vydává MQCONNX a získává sdílený popisovač *h1*
2. Podproces 1 otevře frontu a vydá požadavek na získání pomocí *h1*
3. Podproces 2 vydá požadavek na vložení pomocí *h1*
4. Podproces 3 vydá požadavek na vložení pomocí *h1*
5. Problémy podprocesů 2 MQDISC s použitím *h1*

Zatímco manipulátor je používán libovolným vláknem, přístup k připojení není k dispozici pro ostatní podprocesy. Za okolností, kdy je přijatelné, aby podproces čekal na dokončení nějaké předchozí volání z jiného podprocesu, použijte příkaz MQCONNX s volbou MQCNO_HANDLE_SHARE_BLOCK.

Blokování však může způsobit potíže. Předpokládejme, že v kroku “2” na stránce 711vydá vlákno 1 požadavek na získání, který čeká na zprávy, které možná ještě nedorazily (get with wait). V tomto případě zůstanou podprocesy 2 a 3 také čekající (blokované) tak dlouho, jak dlouho trvá požadavek na získání požadavku na vlákno 1. Dáváte-li přednost tomu, aby se volání MQI vrátilo s chybou, pokud již bylo na manipulátoru spuštěno jiné volání MQI, použijte příkaz MQCONNX s volbou MQCNO_HANDLE_SHARE_NO_BLOCK.

Poznámky k použití sdíleného připojení

1. Všechny manipulátory objektů (Hobj) vytvořené otevřením objektu jsou přidruženy k Hconn; takže pro sdílený Hconn jsou objekty Hobjs také sdíleny a použitelné libovolným vláknem pomocí Hconn. Podobně je každá jednotka práce spouštěná pod Hconn asociována s tímto Hconn, takže je tento atribut sdílen všemi podprocesy se sdíleným Hconn.
2. *Jakýkoli* podproces může volat MQDISC k odpojení sdíleného připojení Hconn, nikoli pouze podprocesu, který volal odpovídající MQCONNX. Operace MQDISC ukončí modul Hconn, který ji znepřístupní pro všechny podprocesy.
3. Jeden podproces může použít více sdílených položek Hcons sériově, například pomocí příkazu MQPUT můžete vložit jednu zprávu pod jedním sdíleným Hconn a pak vložit jinou zprávu pomocí jiného sdíleného Hconn, přičemž každá operace bude pod jinou lokální pracovní jednotkou.
4. Sdílené HConns nelze použít v rámci globální transakce.

proměnná prostředí MQCONNX
Tyto informace vám pomohou porozumět různým volbám volání MQCONNX a způsobu použití s hodnotou MQ_CONNECT_TYPE. Všimněte si, že hodnota MQ_CONNECT_TYPE má pouze vliv na vazby typu STANDARD. Pro ostatní vazby je hodnota MQ_CONNECT_TYPE ignorována.

V systémech IBM MQ for IBM i, IBM MQ for Windowsa IBM MQ v systémech UNIX and Linux můžete použít proměnnou prostředí MQ_CONNECT_TYPE v kombinaci s typem vazby zadaným v poli *Options* struktury MQCNO, která se používá u volání MQCONNX.

Tabulka 99. Proměnná prostředí MQ_CONNECT_TYPE

Volba volání MQCONNX	**proměnná prostředí MQ_CONNECT_TYPE**	**Výsledek**
STANDARD	Nedefinovaný	STANDARD

Tabulka 99. Proměnná prostředí MQ_CONNECT_TYPE (pokračování)		
Volba volání MQCONNX	**proměnná prostředí MQ_CONNECT_TYPE**	**Výsledek**
STANDARD	STANDARD	STANDARD
STANDARD	Rychlý	STANDARD
STANDARD	CLIENT	CLIENT
STANDARD	LOKÁLNÍ	STANDARD

Není-li parametr MQCNO_STANDARD_BINDING zadán, můžete použít parametr MQCNO_NONE, který bude standardně zobrazovat MQCNO_STANDARD_BINDING.

Odpojení programů od správce front pomocí příkazu MQDISC

Tato část obsahuje informace o odpojování programů od správce front pomocí funkce MQDISC.

Když program, který se připojil ke správci front pomocí volání MQCONN nebo MQCONNX, dokončil veškerou interakci se správcem front, přeruší spojení pomocí volání MQDISC s výjimkou následujících:

- Na serveru CICS Transaction Server pro aplikace z/OS , kde je volání volitelné, pokud nebyl použit MQCONNX a chcete před ukončením aplikace zrušit značku připojení.
- V systému IBM MQ for IBM i , kde se při odhlášení z operačního systému provede implicitní volání MQDISC.

Jako vstup do volání MQDISC je třeba při připojení ke správci front dodat manipulátor připojení (Hconn), který byl vrácen příkazem MQCONN nebo MQCONNX.

S výjimkou produktu CICS v systému z/OSje po volání funkce MQDISC volání popisovače připojení (Hconn) již neplatné a nelze znovu vydat žádná další volání MQI, dokud znovu nezavoláte MQCONN nebo MQCONNX. MQDISC provádí implicitní MQCLOSE pro všechny objekty, které jsou stále otevřené pomocí tohoto ovladače.

z/OS U klienta připojeného k produktu z/OSdojde při volání MQDISC k implicitnímu provedení operace commit, ale všechny manipulátory front, které jsou stále otevřené, se nezavřou, dokud kanál skutečně neskončí.

Použijete-li produkt MQCONNX k připojení k produktu IBM MQ for z/OS, funkce MQDISC také ukončí rozsah značky připojení zavedené rozhraním MQCONNX. Pokud však v aplikaci CICS, IMSnebo RRS existuje aktivní jednotka zotavení přidružená ke značce připojení, operace MQDISC je odmítnuta s kódem příčiny MQRC_CONN_TAG_NOT_RELEASED.

Popisy parametrů jsou uvedeny v popisu volání MQDISC v MQDISC.

Když není vydáno MQDISC

Při ukončení vytvářeného podprocesu se vyčistí standardní nesdílené připojení (Hconn). Sdílené připojení je implicitně odpojeno a odpojeno až po ukončení celého procesu. Pokud se podproces, který vytvořil sdílený Hconn, ukončí, zatímco Hconn stále existuje, je Hconn stále použitelný.

Kontrola oprávnění

Volání MQCLOSE a MQDISC obvykle neprovádí kontrolu oprávnění.

V normálním průběhu událostí se úloha, která má oprávnění k otevření nebo připojení k objektu IBM MQ , uzavře nebo se odpojí od tohoto objektu. I když je oprávnění úlohy, která se připojila nebo otevřela objekt IBM MQ , odvolána, volání MQCLOSE a MQDISC jsou akceptována.

Otevírání a zavírání objektů

Tyto informace poskytují vhled do otevírání a zavírání objektů IBM MQ .

Chcete-li provést některou z následujících operací, je třeba nejprve *otevřít* příslušný objekt IBM MQ :

- Vložení zpráv do fronty
- Získat (procházet nebo načíst) zprávy z fronty
- Nastavit atributy objektu
- Dotaz na atributy libovolného objektu

Použijte volání MQOPEN k otevření objektu pomocí voleb volání, abyste uvedli, co chcete dělat s objektem. Jedinou výjimkou je, že chcete-li vložit jednu zprávu do fronty, a ihned zavřít frontu. V takovém případě můžete vynechat fázi *otevírání* pomocí volání MQPUT1 (viz “Vložení jedné zprávy do fronty pomocí volání MQPUT1” na stránce 731).

Před otevřením objektu pomocí volání MQOPEN je třeba program připojit ke správci front. To je podrobně vysvětleno pro všechna prostředí v produktu “Připojování k správci front a odpojování od něj” na stránce 704.

Existují čtyři typy objektů IBM MQ , které lze otevřít:

- Fronta
- Seznam názvů
- Definice procesu
- Správce front

Všechny tyto objekty otevírejte podobným způsobem pomocí volání MQOPEN. Další informace o objektech IBM MQ naleznete v tématu Typy objektů.

Stejný objekt můžete otevřít více než jednou, a pokaždé, když získáte nový popisovač objektu. Možná budete chtít procházet zprávy ve frontě pomocí jednoho úchytu a odebrat zprávy ze stejné fronty s použitím jiného ovladače. To šetří využití prostředků k uzavření a opětnému otevření téhož objektu. Můžete také otevřít frontu pro procházení *a* odebíráním zpráv současně.

Kromě toho můžete otevřít více objektů s jedinou operací MQOPEN a zavřít je pomocí příkazu MQCLOSE. Informace o tom, jak to provést, viz “Distribuční seznamy” na stránce 732 .

Při pokusu o otevření objektu správce front zkontroluje, zda máte autorizaci k otevření daného objektu pro volby, které jste zadali v rámci volání MQOPEN.

Objekty jsou automaticky zavřeny, když se program odpojí od správce front. V prostředí IMS je odpojení vynuceno, když program zahajuje zpracování pro nového uživatele po volání GU (get unique) IMS . Na platformě IBM i jsou objekty automaticky zavřeny při ukončení úlohy.

Je dobrým programovacím zvykem zavřít objekty, které jste otevřeli. Použijte volání MQCLOSE k provedení tohoto.

Chcete-li zjistit více o otevírání a zavírání objektů, použijte následující odkazy:

- “Otevírání objektů pomocí volání MQOPEN” na stránce 714
- “Vytváření dynamických front” na stránce 721
- “Otevírání vzdálených front” na stránce 722
- “Zavírání objektů pomocí volání MQCLOSE” na stránce 722

Související pojmy

“Přehled rozhraní fronty zpráv” na stránce 690
Zde jsou uvedeny informace o komponentách rozhraní MQI (Message Queue Interface).

“Připojování k správci front a odpojování od něj” na stránce 704
Chcete-li používat programovací služby IBM MQ , musí mít program připojení ke správci front. V této části se dozvíte, jak se připojit k správci front a odpojit je od správce front.

“Vložení zpráv do fronty” na stránce 723
V této části se dozvíte, jak vložit zprávy do fronty.

“Získávání zpráv z fronty” na stránce 737

Tyto informace použijte k získání informací o získávání zpráv z fronty.

“Inquaning about a nastavení atributů objektu” na stránce 815
Atributy jsou vlastnosti, které definují charakteristiky objektu IBM MQ .

“Potvrzení a zálohování jednotek práce” na stránce 818
Tyto informace popisují, jak potvrdit a zazálohovat všechny zotavitelné operace get a put, které se vyskytly v jednotce práce.

“Spuštění aplikací produktu IBM MQ pomocí spouštěčů” na stránce 829
Informace o spouštěčích a o tom, jak spustit aplikace IBM MQ pomocí spouštěčů.

“Práce s MQI a klastry” na stránce 847
Existují speciální volby na voláních a návratových kódech, které se vztahují ke klastrování.

“Použití a zápis aplikací v systému IBM MQ for z/OS” na stránce 851
Aplikace produktu IBM MQ for z/OS mohou být vytvořeny z programů spuštěných v mnoha různých prostředích. To znamená, že mohou využít výhody zařízení dostupných ve více než jednom prostředí.

“Aplikace mostu IMS a IMS v systému IBM MQ for z/OS” na stránce 57
Tyto informace vám pomohou při psaní aplikací produktu IMS pomocí produktu IBM MQ.

Otevírání objektů pomocí volání MQOPEN

Tyto informace použijte k seznámení se s otevíráním objektů pomocí volání MQOPEN.

Vstup do volání MQOPEN je třeba zadat:

- Popisovač připojení. Pro aplikace produktu CICS v systému z/OSmůžete zadat konstantu MQHC_DEF_HCONN (má hodnotu nula) nebo použít obslužnou rutinu připojení vrácenou voláním MQCONN nebo MQCONNX. Pro ostatní programy vždy použijte obslužnou rutinu připojení vrácenou voláním MQCONN nebo MQCONNX.
- Popis objektu, který chcete otevřít, s použitím struktury deskriptoru objektu (MQOD).
- Jedna nebo více voleb, které řídí akci volání.

Výstup z MQOPEN je:

- Popisovač objektu, který představuje váš přístup k objektu. Tato hodnota se používá při zadávání veškerých následných volání MQI.
- Upravená struktura popisovače objektu, pokud vytváříte dynamickou frontu (a je podporována na vaší platformě).
- Kód dokončení.
- Kód příčiny.

Rozsah manipulátoru objektu

Rozsah manipulátoru objektu (Hobj) je stejný jako rozsah platnosti manipulátoru připojení (Hconn).

To je zahrnuto v “Rozsah MQCONN nebo MQCONNX” na stránce 707 a “Sdílená připojení (nezávislá na podprocesech) s MQCONNX” na stránce 710. V některých prostředích však existují další pokyny:

CICS
V programu CICS můžete použít popisovač pouze v rámci stejné úlohy CICS , ze které jste provedli volání MQOPEN.

Dávka IMS a z/OS
V prostředí IMS a v dávkových prostředích můžete použít popisovač v rámci stejné úlohy, ale ne v rámci žádných dílčích úloh.

Popisy parametrů volání MQOPEN jsou uvedeny v části MQOPEN.

Následující oddíly popisují informace, které je třeba dodat jako vstup pro funkci MQOPEN.

Označení objektů (struktura MQOD)

K identifikaci objektu, který chcete otevřít, použijte strukturu MQOD. Tato struktura je vstupním parametrem pro volání MQOPEN. (Strukturu upraví správce front, když použijete volání MQOPEN k vytvoření dynamické fronty.)

Podrobné informace o struktuře MQOD naleznete v části MQOD.

Informace o použití struktury MQOD pro distribuční seznamy viz “Použití struktury MQOD” na stránce 733 v části “Distribuční seznamy” na stránce 732.

Rozpoznání názvu
Jak volání MQOPEN řeší názvy front a správců front.

Poznámka: Alias správce front je definice vzdálené fronty bez pole RNAME .

Když otevřete frontu IBM MQ , volání MQOPEN provede funkci rozpoznání názvu ve vámi uvedeném názvu fronty. To určuje, jakou frontu provádí správce front následné operace. To znamená, že když uvedete název alias fronty nebo vzdálené fronty v deskriptoru objektu (MQOD), volání vyřeší název buď do lokální fronty, nebo do přenosové fronty. Je-li fronta otevřena pro libovolný typ vstupu, procházení nebo nastavení, interpretuje se jako lokální fronta, pokud existuje, a selže, pokud tam není. Překládá se na nelokální frontu pouze tehdy, je-li otevřena pouze pro výstup, dotazuje se pouze nebo pouze na výstup a dotazuje se. Přehled procesu rozlišování názvů naleznete v tématu Tabulka 100 na stránce 715 . Název, který jste zadali v produktu *ObjectQMgrName* , je vyřešen *před* , než je uvedeno v *ObjectName*.

Tabulka 100 na stránce 715 také ukazuje, jak můžete použít lokální definici vzdálené fronty k definování aliasu pro název správce front. To vám umožní určit, která přenosová fronta se použije při vkládání zpráv do vzdálené fronty, takže byste mohli například použít jednu přenosovou frontu pro zprávy určené pro mnoho vzdálených správců front.

Chcete-li použít následující tabulku, nejprve si přečtěte dva levé sloupce pod záhlavím **Vstup do MQOD**a vyberte příslušný případ. Poté si přečtěte všechny pokyny v příslušném řádku. Podle pokynů ve sloupcích **Vyřešené názvy** se můžete vrátit buď do sloupců **Vstup do MQOD** a vložit hodnoty podle pokynů, nebo můžete tabulku ukončit s dodanými výsledky. Můžete být například vyzváni k zadání *ObjectName*.

Tabulka 100. Řešení názvů front při použití funkce MQOPEN

Vstup pro MQOD	**Vstup pro MQOD**	**Převedené názvy**	**Převedené názvy**	**Převedené názvy**
ObjectQMgrName	*ObjectName*	*ObjectQMgrName*	*ObjectName*	Transmission queue
Prázdný nebo lokální správce front	Lokální fronta bez atributu CLUSTER	Lokální správce front	Zadejte *ObjectName*	Nehodí se (lokální používaná fronta)
Prázdný správce front	Lokální fronta s atributem CLUSTER	Vybraný správce front klastru vybraného klastru správy pracovní zátěže nebo specifický správce front klastru vybraný na operaci PUT	Zadejte *ObjectName*	SYSTEM.CLUSTER.TRANSMIT.QUEUE a použitá lokální fronta SYSTEM.QSG.TRANSMIT.QUEUE (viz poznámka)
Lokální správce front	Lokální fronta s atributem CLUSTER	Lokální správce front	Zadejte *ObjectName*	Nehodí se (lokální používaná fronta)
Prázdný nebo lokální správce front	Modelová fronta	Lokální správce front	Generovaný název	Nehodí se (lokální používaná fronta)

Tabulka 100. Řešení názvů front při použití funkce MQOPEN (pokračování)

Vstup pro MQOD	**Vstup pro MQOD**	**Převedené názvy**	**Převedené názvy**	**Převedené názvy**
Prázdný nebo lokální správce front	Alias fronty s atributem CLUSTER nebo bez ní	Znovu proveďte interpretaci názvu s názvem *ObjectQMgrNázev* a zadejte vstupní hodnotu *ObjectName* do hodnoty *BaseQName* v objektu definice alias fronty. Nesmí se interpretovat jako lokálně definovaný alias, kde je zadán *ObjectQMgrNázev* , ale může se převést na klastrovaný alias (hostovaný na jiných správcích front), kde je *ObjectQMgrNázev* prázdný.		
Lokální správce front	Alias fronty s atributem CLUSTER	Alias nesmí být převáděn na frontu klastru, která není lokálně definována, nebo pro frontu klastru obsahující stejnou hodnotu *ObjectName* jako alias.		
Prázdný správce front	Alias fronty s atributem CLUSTER	Alias se může interpretovat jako fronta klastru se stejnou hodnotou *ObjectName* jako alias.		
Prázdný nebo lokální správce front	lokální definice vzdálené fronty	Znovu proveďte řešení názvu s parametrem *ObjectQMgrName* nastaveným na hodnotu *RemoteQMgrNamea Obj ectName* nastavenou na *RemoteQName*. Nesmí se interpretovat vzdálené fronty		Název atributu *XmitQName* , je-li jiný než blank; v opačném případě *RemoteQMgrName* v objektu definice vzdálené fronty. SYSTEM.QSG.TRANSMI T.QUEUE (viz poznámka)
Prázdný správce front	Nebyl nalezen odpovídající lokální objekt; byla nalezena fronta klastru	Vybraný správce front klastru vybraného klastru správy pracovní zátěže nebo specifický správce front klastru vybraný na operaci PUT	Zadejte *ObjectName*	SYSTEM.CLUSTER.TRAN SMIT.QUEUE SYSTEM.QSG.TRANSMI T.QUEUE (viz poznámka)

Tabulka 100. Řešení názvů front při použití funkce MQOPEN (pokračování)

Vstup pro MQOD	**Vstup pro MQOD**	**Převedené názvy**	**Převedené názvy**	**Převedené názvy**
Prázdný nebo lokální správce front	Nebyl nalezen odpovídající lokální objekt; fronta klastru nebyla nalezena		Chyba, fronta nebyla nalezena	Nelze použít
Název správce front ve stejné skupině sdílení front jako lokální správce front	Lokální sdílená fronta	Lokální správce front	Zadejte *ObjectName*	Nelze použít
Název lokální přenosové fronty	(Nevyřešeno)	Zadejte *ObjectQMgrNázev*	Zadejte *ObjectName*	Zadejte *ObjectQMgrNázev* SYSTEM.QSG.TRANSMIT.QUEUE (viz poznámka)
Definice alias správce front (*RemoteQMgrName* může být lokální správce front)	(Nevyřešeno, vzdálená fronta)	Znovu proveďte řešení názvu s parametrem *ObjectQMgrName* nastaveným na hodnotu *RemoteQMgrName*. Musí se interpretovat na vzdálené fronty	Zadejte *ObjectName*	Název atributu *XmitQName* , je-li jiný než blank; v opačném případě *RemoteQMgrName* v objektu definice vzdálené fronty. SYSTEM.QSG.TRANSMIT.QUEUE (viz poznámka)
Správce front není názvem žádného lokálního objektu; byl nalezen správce front klastru nebo alias správce front.	(Nevyřešeno)	*ObjectQMgrNázev* nebo specifický správce front klastru vybraný na PUT	Zadejte *ObjectName*	SYSTEM.CLUSTER.TRANSMIT.QUEUE SYSTEM.QSG.TRANSMIT.QUEUE (viz poznámka)
Správce front není názvem žádného lokálního objektu; nebyly nalezeny žádné objekty klastru	(Nevyřešeno)	Zadejte *ObjectQMgrNázev*	Zadejte *ObjectName*	Atribut *DefXmitQName* správce front, ve kterém je podporován *DefXmitQName* . SYSTEM.QSG.TRANSMIT.QUEUE (viz poznámka)

Notes:

1. *BaseQName* je název základní fronty z definice alias fronty.
2. *RemoteQName* je název vzdálené fronty z lokální definice vzdálené fronty.
3. *RemoteQMgrName* je název vzdáleného správce front z lokální definice vzdálené fronty.
4. *XmitQName* je název přenosové fronty z lokální definice vzdálené fronty.

5. Při použití správců front produktu IBM MQ for z/OS , kteří jsou součástí skupiny sdílení front (QSG), lze místo názvu lokálního správce front v produktu Tabulka 100 na stránce 715místo názvu lokální správce front použít název skupiny sdílení front.

   Pokud lokální správce front nemůže otevřít cílovou frontu nebo vložit do fronty zprávu, bude zpráva předána do určeného názvu ObjectQMgrprostřednictvím front v rámci skupiny nebo kanálu produktu IBM MQ .
6. Ve sloupci *ObjectName* tabulky se KLASTR odkazuje na atributy CLUSTER a CLUSNL fronty.
7. SYSTEM.QSG.TRANSMIT.QUEUE se používá v případě, že se lokální a vzdálené správci front nacházejí ve stejné skupině sdílení front; řazení do fronty v rámci skupiny je povoleno.
8. Pokud jste každému klastru odesílacímu kanálu přiřadili jinou přenosovou frontu klastru, SYSTEM.CLUSTER.TRANSMIT.QUEUE nemusí být název přenosové fronty klastru. Další informace o více přenosových frontách klastru najdete v tématu Klastrování: Plánování konfigurace přenosových front klastru.
9. V situaci, kdy správce front není názvem žádného lokálního objektu, správce front klastru nebo alias správce front, který byl nalezen.

   Pokud jste zadali název správce front pomocí produktu **ObjectQMgrName**a existuje více kanálů klastru s různými názvy klastrů, které jsou známy lokálním správcem front, které by dosáhly místa určení, může být kterýkoli z těchto kanálů použit k přesunutí zprávy bez ohledu na název klastru cílové fronty.

   To může být neočekávané, pokud jste očekávali zprávy pro tuto frontu pouze k odeslání prostřednictvím kanálu se stejným názvem klastru jako fronta.

   Produkt **ObjectQMgrName** však v tomto případě má přednost a vyrovnávání pracovní zátěže klastru bere v úvahu všechny kanály, které se mohou dostat k tomuto správci front, bez ohledu na název klastru, ve kterém se nacházejí.

Při otevření fronty aliasu se také otevře základní fronta, do níž je alias převeden, a při otevření vzdálené fronty se také otevře přenosová fronta. Proto nemůžete odstranit frontu, kterou jste určili, nebo frontu, na kterou se rozlišuje, zatímco druhá je otevřená.

Fronta aliasů se sice nemůže přeložit na jinou lokálně definovanou alias frontu (sdílenou v klastru nebo ne), takže je možné ji přeložit na vzdáleně definovanou frontu aliasů klastru, a proto může být zadána jako základní fronta.

Vyřešený název fronty a vyřešený název správce front jsou uloženy v polích *ResolvedQName* a *ResolvedQMgrName* v MQOD.

Další informace o rozlišení názvu v distribuovaném prostředí fronty najdete v tématu Co je to rozlišení názvu fronty?.

Použití voleb volání MQOPEN

V parametru **Options** volání MQOPEN musíte vybrat jednu nebo více voleb pro řízení přístupu, který jste dostali k objektu, který otevíráte. S těmito volbami můžete:

- Otevřete frontu a určete, že všechny zprávy zařazené do této fronty musí být směrovány do stejné instance.
- Chcete-li povolit vkládání zpráv do fronty, otevřete ji.
- Otevřete frontu, abyste mohli procházet zprávy na této frontě
- Otevřete frontu, abyste v něm mohli odebírat zprávy
- Otevřít objekt, který vám umožní dotazovat se na atributy a nastavit jeho atributy (ale můžete nastavit pouze atributy front)
- Otevřít téma nebo řetězec tématu pro publikování zpráv v rámci tohoto tématu
- Přidružit informace o kontextu ke zprávě
- Nominovat alternativní identifikátor uživatele, který má být použit pro kontroly zabezpečení
- Řídit volání, je-li správce front ve stavu uvedení do klidového stavu

Volba MQOPEN pro frontu klastru
Vazba použitá pro popisovač fronty je převzata z atributu fronty produktu **DefBind** , který může převzít hodnotu MQBND_BIND_ON_OPEN, MQBND_BIND_NOT_FIXEDnebo MQBND_BIND_ON_GROUP.

Chcete-li směrovat všechny zprávy do fronty pomocí produktu MQPUT do stejného správce front stejnou přenosovou cestou, použijte volbu MQOO_BIND_ON_OPEN na volání MQOPEN .

Chcete-li určit, že má být místo určení vybráno v čase MQPUT, tj. na základě zpráv po jednotlivých zprávách, použijte volbu MQOO_BIND_NOT_FIXED ve volání MQOPEN .

Chcete-li uvést, že všechny zprávy v skupinách zpráv vložených do fronty s použitím MQPUT jsou alokovány ke stejné cílové instanci, použijte volbu MQOO_BIND_ON_GROUP na volání MQOPEN .

Musí být zadán buď MQOO_BIND_ON_OPEN nebo MQOO_BIND_ON_GROUP , když používáte skupiny zpráv s klastry, aby se zajistilo, že všechny zprávy ve skupině se zpracují ve stejném cíli.

Pokud nezadáte žádnou z těchto voleb, bude použita výchozí hodnota MQOO_BIND_AS_Q_DEF.

Zadáte-li název správce front v produktu MQOD, bude vybrána fronta v daném správci front. Je-li název správce front prázdný, lze vybrat libovolnou instanci. Další informace viz “MQOPEN a klastry” na stránce 848.

Pokud otevřete klastrovou frontu pomocí definice QALIAS , jsou některé atributy fronty definovány frontou aliasů a nikoli základní frontou. Atributy klastru jsou mezi atributy definice základní fronty, která je přepsána frontou aliasů. Například v následujícím úseku kódu je fronta klastru otevřena s produktem MQOO_BIND_NOT_FIXED a ne s produktem MQOO_BIND_ON_OPEN. Definice fronty klastru je zveřejňována v celém klastru, definice alias fronty je lokální vzhledem ke správci front.

```
DEFINE QLOCAL(CLQ1) CLUSTER(MYCLUSTER) DEFBIND(OPEN) REPLACE
DEFINE QALIAS(ACLQ1) TARGQ(CLQ1) DEFBIND(NOTFIXED) REPLACE
```

Volba MQOPEN pro vložení zpráv
Chcete-li otevřít frontu nebo téma pro vložení zpráv do fronty, použijte volbu MQOO_OUTPUT.

Volba MQOPEN pro procházení zpráv
Chcete-li otevřít frontu, aby bylo možné *procházet* zprávy, použijte volání MQOPEN s volbou MQOO_BROWSE.

Tím se vytvoří *kurzor procházení* , který správce front používá k identifikaci další zprávy ve frontě. Další informace viz “Procházení zpráv ve frontě” na stránce 769.

Poznámka:

1. Zprávy ve vzdálené frontě nelze procházet, neotvírejte vzdálenou frontu pomocí volby MQOO_BROWSE.
2. Při otevírání rozdělovníku nelze tuto volbu uvést. Další informace o distribučních seznamech viz “Distribuční seznamy” na stránce 732.
3. Funkci MQOO_CO_OP spolu s aplikací MQOO_BROWSE použijte v případě, že používáte spolupracující procházení, viz Volby .

Volby MQOPEN pro odebrání zpráv
K odebrání zpráv z fronty slouží tři volby řízení otevírání fronty.

V libovolném volání MQOPEN můžete použít pouze jeden z nich. Tyto volby definují, zda má váš program výhradní nebo sdílený přístup do fronty. *Výlučný přístup* znamená, že dokud nezavřete frontu, z ní můžete odebírat pouze zprávy. Pokud se jiný program pokusí otevřít frontu za účelem odebrání zpráv, volání MQOPEN se nezdaří. *Sdílený přístup* znamená, že více než jeden program může odebrat z fronty.

Nejvhodnější metodou je přijetí typu přístupu, který byl určen pro frontu, když byla fronta definována. Definice fronty zahrnuje nastavení *Shareability* a Atributy produktu **DefInputOpenOption** . Chcete-li tento přístup přijmout, použijte volbu MQOO_INPUT_AS_Q_DEF. Podívejte se na Tabulka 101 na stránce 720 , abyste zjistili, jak nastavení těchto atributů ovlivňuje typ přístupu, který bude poskytnut při použití této volby.

Tabulka 101. Jak atributy fronty a volby volání MQOPEN mají vliv na přístup k frontám				
Atributy fronty		**Typ přístupu s volbami MQOPEN**		
Shareability	*DefInputOpenOption*	**AS Q_DEF**	**SHARED**	**EXCLUSIVE**
Možnost sdílení	SHARED	sdíleno	sdíleno	výlučný
Možnost sdílení	EXCLUSIVE	výlučný	sdíleno	výlučný
NESDÍLET *	SDÍLENO *	výlučný	výlučný	výlučný
NESDÍLET	EXCLUSIVE	výlučný	výlučný	výlučný

Poznámka: * Ačkoli můžete definovat frontu pro použití této kombinace atributů, je výchozí vstupní otevřená volba potlačena atributem sdílené schopnosti.

Alternativně:

- Pokud víte, že vaše aplikace může úspěšně pracovat i v případě, že jiné programy mohou zprávy z fronty odstranit současně, použijte volbu MQOO_INPUT_SHARED. Tabulka 101 na stránce 720 ukazuje, jak v některých případech budete mít výlučný přístup do fronty, a to i s touto volbou.
- Pokud víte, že vaše aplikace může fungovat úspěšně pouze v případě, že jiným programům není zabráněno v odstraňování zpráv z fronty současně, použijte volbu MQOO_INPUT_EXCLUSIVE.

Poznámka:

1. Zprávy ze vzdálené fronty nelze odebrat. Proto nelze vzdálenou frontu otevřít pomocí žádné z voleb MQOO_INPUT_ *.
2. Při otevírání rozdělovníku nelze tuto volbu uvést. Další informace uvádí téma "Distribuční seznamy" na stránce 732.

Volby MQOPEN pro nastavení a zjišťování informací o atributech
Chcete-li otevřít frontu tak, abyste mohli nastavit její atributy, použijte volbu MQOO_SET.

Nemůžete nastavit atributy žádného jiného typu objektu (viz "Inquaning about a nastavení atributů objektu" na stránce 815).

Chcete-li otevřít objekt tak, abyste se mohli dotázat na jeho atributy, použijte volbu MQOO_INQUIRE.

Poznámka: Při otevírání rozdělovníku nelze tuto volbu uvést.

Volby MQOPEN vztahující se ke kontextu zprávy
Chcete-li mít možnost přidružit kontextové informace ke zprávě, když ji vložíte do fronty, musíte při otevření fronty použít jednu z voleb kontextu zprávy.

Volby vám umožňují rozlišovat mezi informacemi o kontextu, které souvisí s *uživatelem* , který je původcem zprávy, a to, které se vztahuje k *aplikaci* , která je původcem zprávy. Můžete také zvolit nastavení kontextové informace při vložení zprávy do fronty nebo můžete zvolit přepnutí kontextu automaticky z jiného manipulátorů fronty.

Související pojmy
"kontext zprávy" na stránce 41
Informace *Kontext zprávy* umožňují aplikaci, která načte zprávu, zjistit informace o odesílateli zprávy.

"Řízení informací o kontextu zprávy" na stránce 729
Použijete-li volání MQPUT nebo MQPUT1 k vložení zprávy do fronty, můžete určit, že správce front má přidat některé výchozí kontextové informace do deskriptoru zpráv. Aplikace, které mají odpovídající úroveň oprávnění, mohou přidávat další informace o kontextu. Pole voleb ve struktuře MQPMO můžete použít k řízení informací o kontextu.

Volba MQOPEN pro alternativní oprávnění uživatele
Pokusíte-li se otevřít objekt pomocí volání MQOPEN, správce front zkontroluje, zda máte oprávnění k otevření daného objektu. Pokud nemáte oprávnění, volání selže.

Serverová programy však mohou chtít správce front zkontrolovat autorizaci uživatele, pro kterého pracují, spíše než vlastní autorizace serveru. K tomu musí použít volbu MQOO_ALTERNATE_USER_AUTHORITY pro volání MQOPEN a určit alternativní ID uživatele v poli *AlternateUserId* struktury MQOD. Obvykle server získá ID uživatele z informací o kontextu ve zprávě, kterou zpracovává.

z/OS *Volba MQOPEN pro uvedení správce front do klidového stavu*

Pokud při uvedení správce front do klidového stavu použijete volání MQOPEN, může dojít k selhání volání v závislosti na prostředí, které používáte.

Pokud v prostředí CICS v produktu z/OSpoužijete volání MQOPEN, je-li správce front ve stavu uvedení do klidového stavu, volání se vždy nezdaří.

V jiných prostředích z/OS , IBM i, Windows a v prostředí systémů UNIX and Linux se volání nezdaří, je-li správce front uváděn do klidového stavu pouze v případě, že použijete volbu MQOO_FAIL_IF_QUIESCING volání MQOPEN.

Volba MQOPEN pro interpretaci názvů lokálních front

Když otevřete lokální, alias nebo modelovou frontu, vrátí se lokální fronta.

Když však otevřete vzdálenou frontu nebo frontu klastru, pole *ResolvedQName* a *ResolvedQMgrName* struktury MQOD jsou vyplněny názvy vzdálené fronty a vzdáleného správce front nalezeného v definici vzdálené fronty nebo ve zvolené vzdálené frontě klastru.

Použijte volbu MQOO_RESOLVE_LOCAL_Q volání MQOPEN k vyplnění hodnoty *ResolvedQName* ve struktuře MQOD s názvem lokální fronty, která byla otevřena. *ResolvedQMgrName* je podobně zaplněn názvem lokálního správce front, který je hostitelem lokální fronty. Toto pole je dostupné pouze u verze 3 struktury MQOD; je-li struktura menší než verze 3, je MQOO_RESOLVE_LOCAL_Q ignorována, aniž by byla vrácena chyba.

Určíte-li hodnotu MQOO_RESOLVE_LOCAL_Q, je-li například otevřena vzdálená fronta, *ResolvedQName* je název přenosové fronty, do které budou zprávy vloženy. *ResolvedQMgrName* je název lokálního správce front, který je hostitelem přenosové fronty.

Vytváření dynamických front

Dynamickou frontu použijte, když nepotřebujete frontu po ukončení aplikace.

Můžete například použít dynamickou frontu pro svou frontu pro odpověď. Název fronty pro odpověď zadejte do pole *ReplyToQ* struktury MQMD, když vložíte zprávu do fronty (viz "Definování zpráv pomocí struktury MQMD" na stránce 724).

Chcete-li vytvořit dynamickou frontu, použijte šablonu známou jako modelovou frontu spolu s voláním MQOPEN. Frontu modelu vytvoříte pomocí příkazů IBM MQ nebo operací a řídicích panelů. Dynamická fronta, kterou vytvoříte, převezme atributy modelové fronty.

Když voláte MQOPEN, zadejte název modelové fronty do pole *ObjectName* struktury MQOD. Po dokončení volání je pole *ObjectName* nastaveno na název dynamické fronty, která je vytvořena. Pole *ObjectQMgrName* je také nastaveno na název lokálního správce front.

Název dynamické fronty, kterou vytvoříte, můžete zadat třemi způsoby:

- Zadejte celé jméno, které chcete v poli *DynamicQName* struktury MQOD.
- Uveďte předponu (méně než 33 znaků) pro název a umožněte správci front generovat zbývající část názvu. To znamená, že správce front vygeneruje jedinečný název, ale stále máte nějaký ovládací prvek (například můžete chtít, aby každý uživatel používal určitou předponu, nebo můžete chtít udělit speciální klasifikaci zabezpečení pro fronty s určitou předponou v jejich jménu). Chcete-li použít tuto metodu, uveďte hvězdičku (*) pro poslední neprázdný znak pole *DynamicQName* . Neuvádějte pro název dynamické fronty jednu hvězdičku (*).
- Povolit správci front generovat úplný název. Chcete-li použít tuto metodu, uveďte hvězdičku (*) v první znakové pozici pole *DynamicQName* .

Další informace o těchto metodách naleznete v popisu pole DynamicQName .

Existují další informace o dynamických frontách v Dynamické a modelové frontě.

Otevírání vzdálených front

Vzdálená fronta je fronta, jejímž vlastníkem je jiný správce front, než ten, ke kterému je aplikace připojena.

Chcete-li otevřít vzdálenou frontu, použijte volání MQOPEN jako pro lokální frontu. Název fronty můžete zadat následujícím způsobem:

1. V poli *ObjectName* struktury MQOD zadejte název vzdálené fronty, jak je známo, do správce front *local* .

   Poznámka: V tomto případě ponechte pole *ObjectQMgrName* prázdné.
2. V poli *ObjectName* struktury MQOD zadejte název vzdálené fronty, jak je známo ve správci front *remote* . Do pole *ObjectQMgrName* zadejte:

   - Název přenosové fronty, která má stejný název jako vzdálený správce front. Název a velikost písmen (velká písmena, malá písmena nebo směs) se musí přesně shodovat s *přesně*.
   - Název objektu aliasu správce front, který je interpretovaný jako správce cílové fronty nebo pro přenosovou frontu.

   To sděluje správci front místo určení zprávy a také přenosovou frontu, kterou je třeba pro získání této zprávy umístit.
3. Je-li v poli *ObjectName* struktury MQOD podporováno *DefXmitQname* , zadejte název vzdálené fronty, jak je znám ve *vzdáleném* správci front.

   Poznámka: Nastavte pole *ObjectQMgrName* na název vzdáleného správce front (v tomto případě nemůže být ponecháno prázdné).

Při volání MQOPEN jsou ověřovány pouze lokální názvy. Poslední kontrola existence přenosové fronty, která má být použita.

Tyto metody jsou shrnuty vTabulka 100 na stránce 715.

Zavírání objektů pomocí volání MQCLOSE

Chcete-li objekt zavřít, použijte volání MQCLOSE.

Je-li objektem fronta, povšimněte si následujících:

- Před zavřením není nutné vyprázdnit dočasnou dynamickou frontu.

  Když zavřete dočasnou dynamickou frontu, odstraní se fronta spolu se všemi zprávami, které se na ní mohou stále nacházet. To platí i v případě, že existují nepotvrzené příkazy MQGET, MQPUT nebo MQPUT1 nevyřízené pro danou frontu.
- Máte-li v produktu IBM MQ for z/OSnějaké požadavky MQGET s nevyřízenou volbou MQGMO_SET_SIGNAL pro tuto frontu, budou zrušeny.
- Pokud jste frontu otevřeli pomocí volby MQOO_BROWSE, je váš kurzor procházení zničen.

Uzavření nesouvisí se synchronizačním bodem, takže můžete zavřít fronty před nebo po bodu synchronizace.

Jako vstup do volání MQCLOSE je třeba dodat:

- Popisovač připojení. Použijte stejný manipulátor připojení, který byl použit k jeho otevření, nebo případně pro aplikace CICS v systému z/OSmůžete zadat konstantu MQHC_DECF_HCONN (která má hodnotu nula).
- Popisovač objektu, který chcete zavřít. Získejte jej z výstupu volání MQOPEN.
- MQCO_NONE v poli *Options* (pokud nezavíráte trvalou dynamickou frontu).
- Pomocí této volby lze určit, zda má správce front frontu odstranit i v případě, že v ní stále existují zprávy (při zavírání trvalé dynamické fronty).

Výstup z MQCLOSE je:

- Kód dokončení
- Kód příčiny

- Popisovač objektu, reset na hodnotu MQHO_UNUSABLE_HOTBJ

Popisy parametrů volání MQCLOSE jsou uvedeny v části MQCLOSE.

Vložení zpráv do fronty

V této části se dozvíte, jak vložit zprávy do fronty.

K vložení zpráv do fronty použijte volání MQPUT. Pomocí opakovaného volání MQPUT lze opakovaně vkládat zprávy do stejné fronty po počátečním volání MQOPEN. Až dokončíte vkládání všech zpráv do fronty, volejte MQCLOSE.

Chcete-li vložit jednu zprávu do fronty a zavřít frontu okamžitě, můžete použít volání MQPUT1 . Příkaz MQPUT1 provádí stejné funkce jako následující posloupnost volání:

- MQOPEN
- MQPUT
- MQCLOSE

Obecně však platí, že pokud máte více než jednu zprávu, která má být vložena do fronty, je efektivnější použít volání MQPUT. Závisí to na velikosti zprávy a na platformě, na které pracujete.

Chcete-li zjistit více o vkládání zpráv do fronty, použijte následující odkazy:

- “Vložení zpráv do lokální fronty pomocí volání MQPUT” na stránce 724
- “Vložení zpráv do vzdálené fronty” na stránce 728
- “Nastavení vlastností zprávy” na stránce 728
- “Řízení informací o kontextu zprávy” na stránce 729
- “Vložení jedné zprávy do fronty pomocí volání MQPUT1” na stránce 731
- “Distribuční seznamy” na stránce 732
- “Některé případy, kdy volání vložení selžou” na stránce 737

Související pojmy

“Přehled rozhraní fronty zpráv” na stránce 690
Zde jsou uvedeny informace o komponentách rozhraní MQI (Message Queue Interface).

“Připojování k správci front a odpojování od něj” na stránce 704
Chcete-li používat programovací služby IBM MQ , musí mít program připojení ke správci front. V této části se dozvíte, jak se připojit k správci front a odpojit je od správce front.

“Otevírání a zavírání objektů” na stránce 712
Tyto informace poskytují vhled do otevírání a zavírání objektů IBM MQ .

“Získávání zpráv z fronty” na stránce 737
Tyto informace použijte k získání informací o získávání zpráv z fronty.

“Inquaning about a nastavení atributů objektu” na stránce 815
Atributy jsou vlastnosti, které definují charakteristiky objektu IBM MQ .

“Potvrzení a zálohování jednotek práce” na stránce 818
Tyto informace popisují, jak potvrdit a zazálohovat všechny zotavitelné operace get a put, které se vyskytly v jednotce práce.

“Spuštění aplikací produktu IBM MQ pomocí spouštěčů” na stránce 829
Informace o spouštěčích a o tom, jak spustit aplikace IBM MQ pomocí spouštěčů.

“Práce s MQI a klastry” na stránce 847
Existují speciální volby na voláních a návratových kódech, které se vztahují ke klastrování.

“Použití a zápis aplikací v systému IBM MQ for z/OS” na stránce 851
Aplikace produktu IBM MQ for z/OS mohou být vytvořeny z programů spuštěných v mnoha různých prostředích. To znamená, že mohou využít výhody zařízení dostupných ve více než jednom prostředí.

“Aplikace mostu IMS a IMS v systému IBM MQ for z/OS” na stránce 57
Tyto informace vám pomohou při psaní aplikací produktu IMS pomocí produktu IBM MQ.

Vložení zpráv do lokální fronty pomocí volání MQPUT

Tyto informace použijte k získání informací o vkládání zpráv do lokální fronty pomocí volání MQPUT.

Jako vstup do volání MQPUT musíte zadat:

- Popisovač připojení (Hconn).
- Popisovač fronty (Hobj).
- Popis zprávy, kterou chcete vložit do fronty. To je ve formě struktury deskriptoru zpráv (MQMD).
- Řídicí informace ve formě struktury voleb put-message (MQPMO).
- Délka dat obsažených ve zprávě (MQLONG).
- Samotná data zprávy.

Výstup z volání MQPUT je následující:

- Kód příčiny (MQLONG)
- Kód dokončení (MQLONG)

Pokud je volání úspěšně dokončeno, vrací také strukturu vaší volby a strukturu deskriptoru zprávy. Volání upravuje strukturu voleb tak, aby zobrazovala název fronty a správce front, do kterého byla zpráva odeslána. Pokud požadujete, aby správce front generoval jedinečnou hodnotu pro identifikátor zprávy, kterou umisťujete (zadáním binární nuly v poli *MsgId* struktury MQMD), volání vloží hodnotu do pole *MsgId* před vrácením této struktury na vás. Tuto hodnotu obnovte, než vydáte další požadavek MQPUT.

V MQPUTje uveden popis volání MQPUT.

Další informace o informacích potřebných jako vstup pro volání MQPUT naleznete v následujících odkazech:

- “Určení manipulátorů” na stránce 724
- “Definování zpráv pomocí struktury MQMD” na stránce 724
- “Určení voleb pomocí struktury MQPMO” na stránce 725
- “Data ve zprávě” na stránce 727
- “Vložení zpráv: Použití obslužných rutin zpráv” na stránce 728

Určení manipulátorů

Pro popisovač připojení (*Hconn*) v produktu CICS v aplikacích produktu z/OS můžete zadat konstantu MQHC_DEF_HCONN (která má hodnotu nula), nebo můžete použít manipulátor připojení vrácený voláním MQCONN nebo MQCONNX. Pro ostatní aplikace vždy použijte obslužnou rutinu připojení vrácenou voláním MQCONN nebo MQCONNX.

V jakémkoli prostředí, ve kterém pracujete, použijte stejný popisovač fronty (*Hobj*), který je vrácen voláním MQOPEN.

Definování zpráv pomocí struktury MQMD

Struktura deskriptoru zpráv (MQMD) je vstupní/výstupní parametr pro volání MQPUT a MQPUT1 . Použijte jej k definování zprávy, kterou vkládáte do fronty.

Je-li MQPRI_PRIORITY_AS_Q_DEF nebo MQPER_PERSISTENCE_AS_Q_DEF zadán pro zprávu a fronta je fronta klastru, použijí se hodnoty fronty, na které se má provést příkaz MQPUT. Je-li tato fronta zakázána pro operaci MQPUT, volání se nezdaří. Další informace naleznete v tématu Konfigurace klastru správce front .

Poznámka: Použijte MQPMO_NEW_MSG_ID a MQPMO_NEW_CORREL_ID před vložením nové zprávy, abyste se ujistili, že jsou *MsgId* a *CorrelId* jedinečné. Hodnoty v těchto polích se vrátí v úspěšném volání MQPUT.

Existuje úvod do vlastností zprávy, které MQMD popisuje v produktu “Zprávy produktu IBM MQ” na stránce 13, a v MQMDje popis struktury samotné.

Určení voleb pomocí struktury MQPMO

Použijte strukturu MQPMO (Vložit volbu zprávy) k předání voleb pro volání MQPUT a MQPUT1 .

Následující oddíly vám pomohou při vyplňování polí této struktury. Popis struktury v MQPMOje uveden v popisu struktury.

Struktura obsahuje následující pole:

- *StrucId*
- *Version*
- *Options*
- *Context*
- *ResolvedQName*
- *ResolvedQMgrName*
- *RecsPresent*
- *PutMsgRecsFields*
- *ResponseRecOffset and ResponseRecPtr*
- *OriginalMsgHandle*
- *NewMsgHandle*
- *Action*
- *PubLevel*

Obsah těchto polí je následující:

StrucId
Identifikuje strukturu jako strukturu voleb vložení zprávy. Jedná se o čtyřznakové pole. Vždy zadejte MQPMO_STRUC_ID.

Verze
Popisuje číslo verze struktury. Výchozí hodnota je MQPMO_VERSION_1. Zadáte-li MQPMO_VERSION_2, můžete použít distribuční seznamy (viz “Distribuční seznamy” na stránce 732). Zadáte-li MQPMO_VERSION_3, můžete použít popisovače zpráv a vlastnosti zprávy. Zadáte-li příkaz MQPMO_CURRENT_VERSION, bude vaše aplikace nastavena vždy tak, aby používala nejnovější úroveň.

Volby
Tento ovládací prvek řídí následující:

- Zda je operace vložení zahrnuta do pracovní jednotky
- Kolik kontextových informací je přidruženo ke zprávě
- Místo, odkud jsou informace o kontextu převzaty
- Zda se volání nezdaří, je-li správce front ve stavu uvedení do klidového stavu
- Zda je seskupení nebo segmentace povolena
- Generování nového identifikátoru zprávy a identifikátoru korelace
- Pořadí, ve kterém jsou zprávy a segmenty vloženy do fronty
- Zda se mají interpretovat názvy lokálních front

Pokud ponecháte pole *Options* nastaveno na výchozí hodnotu (MQPMO_NONE), bude vámi zadaná zpráva mít k sobě přidruženy výchozí kontextové informace.

Také způsob, jakým volání pracuje se synchronizačním body, je určen platformou. Výchozí hodnota řízení synchronizačního bodu je yes v z/OS ; pro jiné platformy, to znamená ne.

Kontext
Tím je uveden název popisovače fronty, ze kterého chcete kopírovat informace o kontextu (je-li to požadováno v poli *Options*).

Úvod do kontextu zprávy viz “kontext zprávy” na stránce 41. Informace o použití struktury MQPMO k řízení informací o kontextu ve zprávě naleznete v tématu “Řízení informací o kontextu zprávy” na stránce 729.

ResolvedQName
Obsahuje jméno (po vyřešení jakéhokoli jména alias) fronty, která byla otevřena pro přijetí zprávy. Toto je výstupní pole.

Název ResolvedQMgr
Obsahuje název (po vyřešení všech názvů alias) správce front, který je vlastníkem fronty v produktu *ResolvedQName*. Toto je výstupní pole.

MQPMO může také obsahovat pole požadovaná pro distribuční seznamy (viz “Distribuční seznamy” na stránce 732). Chcete-li tuto funkci použít, použije se verze 2 struktury MQPMO. To zahrnuje následující pole:

RecsPresent
Toto pole obsahuje počet front v seznamu distribuce; tj. počet záznamů vložení záznamů (MQPMR) a odpovídající záznamy odpovědí (MQRR).

Hodnota, kterou zadáte, může být stejná jako počet záznamů objektů poskytnutých v MQOPEN. Je-li však hodnota nižší než počet záznamů objektu poskytnutých v rámci volání MQOPEN, nebo pokud nezadáte žádné záznamy s vložením zprávy, budou hodnoty front, které nejsou definovány, převzaty z výchozích hodnot poskytnutých deskriptorem zprávy. Také je-li hodnota větší než počet poskytnutých záznamů o objektu, budou nadbytečné záznamy vložení zpráv ignorovány.

Doporučuje se provést jednu z následujících možností:

- Chcete-li obdržet sestavu nebo odpověď z každého místa určení, zadejte stejnou hodnotu, jaká se objevuje ve struktuře MQOR, a použijte pole MQPMR obsahující pole *MsgId* . Buď inicializujte tato pole *MsgId* na nuly, nebo uveďte MQPMO_NEW_MSG_ID.

  Po vložení zprávy do fronty jsou v MQPMRs; k dispozici hodnoty *MsgId* , které správce front vytvořil; můžete je použít k identifikaci, který cíl je přidružen ke každé sestavě nebo k odpovědi.
- Pokud nechcete přijímat sestavy nebo odpovědi, vyberte jednu z následujících možností:

  1. Chcete-li identifikovat cíle, které selžou okamžitě, můžete stále chtít zadat stejnou hodnotu do pole *RecsPresent* , jak se objevuje ve struktuře MQOR, a poskytnout MQRRs pro identifikaci těchto cílů. Nezadávejte žádné příkazy MQPMRs.
  2. Pokud nechcete identifikovat místa určení, která se nezdařila, zadejte do pole *RecsPresent* nulu a neposkytujte MQPMRs ani MQRRs.

Poznámka: Pokud používáte MQPUT1, musí být počet ukazatelů odezvy záznamu odezvy a Offset záznamu odezvy nula.

Úplný popis záznamů vložení zpráv (MQPMR) a záznamů odpovědí (MQRR) naleznete v části MQPMR a MQRR.

PutMsgRecFields
Označuje, která pole se nacházejí v každém záznamu vložení zprávy (MQPMR). Seznam těchto polí najdete v tématu “Použití struktury MQPMR” na stránce 736.

PutMsgRecOffset* a *PutMsgRecPtr
Ukazatelé (obvykle v C) a posuny (obvykle v COBOLu) se používají k adresování záznamů vkládání zpráv (viz “Použití struktury MQPMR” na stránce 736 , kde získáte přehled o struktuře MQPMR).

Pole *PutMsgRecPtr* použijte k uvedení ukazatele na první záznam zprávy Put, nebo pole *PutMsgRecOffset* pro uvedení posunutí prvního záznamu zprávy o vložení. Toto je posun od začátku MQPMO. V závislosti na poli *PutMsgRecFields* zadejte hodnotu, která není null pro *PutMsgRecOffset* nebo *PutMsgRecPtr*.

Posunutí ResponseReca Ptr ResponseRec
Také můžete použít ukazatele a posuny k adresování záznamů odpovědí (další informace o záznamech odpovědí viz “Použití struktury MQRR” na stránce 735).

Do pole *ResponseRecPtr* zadejte ukazatel na první záznam odpovědi nebo pole *ResponseRecOffset* , abyste zadali posun prvního záznamu odezvy. Toto je posun od začátku struktury MQPMO. Zadejte nenull hodnotu buď pro *ResponseRecOffset* , nebo *ResponseRecPtr*.

Poznámka: Pokud používáte MQPUT1 k vložení zpráv do rozdělovníku, *ResponseRecPtr* musí být null nebo nula a *ResponseRecOffset* musí být nula.

Verze 3 struktury MQPMO dále obsahuje následující pole:

OriginalMsgManipulátor
Použití tohoto pole může být závislé na hodnotě pole *Akce* . Pokud vkládáte novou zprávu s přidruženými vlastnostmi zprávy, nastavte toto pole na popisovač zprávy, který jste již dříve vytvořili a nastavte vlastnosti. Pokud předáváte, odpovídáte nebo generujete sestavu jako odpověď na dříve načtenou zprávu, bude toto pole obsahovat odkaz na zprávu dané zprávy.

Popisovač NewMsg
Pokud uvedete *NewMsgHandle*, jakékoli vlastnosti vztahující se ke zpracování vlastností přepisu přidružené k *OriginalMsgHandle*. Další informace viz Akce (MQLONG).

Akce
Prostřednictvím tohoto pole můžete určit typ prováděné operace. Možné hodnoty a jejich významy jsou následující:

NOVÁ HODNOTA MQACTP_NEW
Toto je nová zpráva, která nesouvisí s žádným jiným.

MQACTP_FORWARD
Tato zpráva byla načtena dříve a nyní je předávána.

MQACTP_REPLY
Tato zpráva je odpovědí na dříve načtenou zprávu.

SESTAVA MQACTP_REPORT
Tato zpráva je sestava generovaná jako výsledek dříve načtené zprávy.

Další informace viz Akce (MQLONG).

PubLevel
Je-li tato zpráva publikami, můžete toto pole nastavit, abyste určili, které odběry se mají přijmout. Tato publikace bude přijímat pouze odběry s *SubLevel* , které jsou nižší než nebo rovny této hodnotě. Výchozí hodnota je 9, což je nejvyšší úroveň, což znamená, že odběry s libovolným *SubLevel* mohou tuto publikaci přijmout.

Data ve zprávě

Zadejte adresu vyrovnávací paměti, která obsahuje data, v parametru **Buffer** volání MQPUT. Do dat můžete zahrnout cokoliv, co obsahuje data. Množství dat ve zprávách však ovlivňuje výkon aplikace, která je zpracovává.

Maximální velikost dat je určena následujícím způsobem:

- Atribut **MaxMsgLength** správce front
- Atribut **MaxMsgLength** fronty, na kterou stavíte zprávu
- Velikost každého záhlaví zprávy přidaného produktem IBM MQ (včetně záhlaví s dead-letter, MQDLH a záhlaví distribučního seznamu, MQDH)

Atribut **MaxMsgLength** správce front uchovává velikost zprávy, kterou může správce front zpracovat. Tato hodnota má výchozí hodnotu 100 MB pro všechny produkty IBM MQ ve verzi V6 nebo vyšší.

Chcete-li určit hodnotu tohoto atributu, použijte volání MQINQ u objektu správce front. Pro velké zprávy můžete tuto hodnotu změnit.

Atribut **MaxMsgLength** fronty určuje maximální velikost zprávy, kterou lze vložit do fronty. Pokusíte-li se vložit zprávu o větší velikosti, než je hodnota tohoto atributu, volání MQPUT se nezdaří. Pokud vkládáte zprávu do vzdálené fronty, maximální velikost zprávy, kterou lze úspěšně uložit, je určena atributem **MaxMsgLength** vzdálené fronty, libovolnými intermediačními přijímajícími frontami, které je zpráva umístěna na trase do místa určení, a použitých kanálů.

Pro operaci MQPUT musí být velikost zprávy menší nebo rovna atributu **MaxMsgLength** jak fronty, tak i správce front. Hodnoty těchto atributů jsou nezávislé, ale doporučuje se nastavit *MaxMsgLength* fronty na hodnotu menší nebo rovnou hodnotě, která má správce front.

Produkt IBM MQ přidává informace záhlaví do zpráv za následujících okolností:

- Když vložíte zprávu do vzdálené fronty, produkt IBM MQ přidá do zprávy strukturu záhlaví přenosu (MQXQH). Tato struktura zahrnuje název cílové fronty a jejího vlastníka správce front.
- Pokud produkt IBM MQ nemůže doručit zprávu do vzdálené fronty, pokusí se vložit zprávu do fronty nedoručených zpráv (undelivered-message). Přidává strukturu MQDLH do zprávy. Tato struktura zahrnuje název fronty místa určení a důvod, proč byla zpráva vložena do fronty nedoručených zpráv.
- Chcete-li odeslat zprávu do více cílových front, produkt IBM MQ přidá záhlaví MQDH do zprávy. Popisuje data, která jsou přítomna ve zprávě, která patří do distribučního seznamu, do přenosové fronty. Zvažte tuto volbu při výběru optimální hodnoty pro maximální délku zprávy.
- Je-li zpráva segment nebo zpráva ve skupině, IBM MQ může přidat MQMDE.

Tyto struktury jsou popsány v MQDH a MQMDE.

Pokud vaše zprávy mají maximální povolenou velikost pro tyto fronty, přidání těchto záhlaví znamená, že operace put se nezdaří, protože zprávy jsou nyní příliš velké. Chcete-li snížit pravděpodobnost, že operace put selže, postupujte takto:

- Nastavte velikost zpráv menších než atribut **MaxMsgLength** pro fronty přenosu a fronty nedoručených zpráv. Povolte alespoň hodnotu konstanty MQ_MSG_HEADER_LENGTH (více pro velké distribuční seznamy).
- Ujistěte se, že atribut **MaxMsgLength** fronty nedoručených zpráv je nastaven na stejnou hodnotu jako *MaxMsgLength* správce front, který vlastní frontu nedoručených zpráv.

Atributy pro správce front a konstanty fronty zpráv jsou popsány v tématu Atributy pro správce front.

z/OS Informace o tom, jak jsou obsluhovány nedoručené zprávy v prostředí distribuovaných front, najdete v tématu Nedoručené a nezpracované zprávy.

Vložení zpráv: Použití obslužných rutin zpráv

Ve struktuře MQPMO jsou k dispozici dva popisovače zpráv, *OriginalMsgHandle* a *NewMsgHandle*. Vztah mezi těmito manipulátory zpráv je definován hodnotou pole MQPMO *Akce* .

Podrobné informace naleznete v tématu Akce (MQLONG). Ovladač zprávy nemusí být nutně vyžadován, aby bylo možné zprávu vložit. Jeho účelem je přidružit vlastnosti ke zprávě, takže je zapotřebí pouze v případě, že používáte vlastnosti zprávy.

Vložení zpráv do vzdálené fronty

Chcete-li vložit zprávu do vzdálené fronty (tj. do fronty vlastněné jiným správcem fronty, než je ten, ke kterému je aplikace připojena) místo lokální fronty, je jedinou přebytečnou otázkou způsob, jak zadat název fronty při otevření.Tento popis je popsán v tématu “Otevírání vzdálených front” na stránce 722. Není zde žádná změna způsobu použití volání MQPUT nebo MQPUT1 pro lokální frontu.

Další informace o používání vzdálených a přenosových front najdete v tématu Distribuované techniky front systému IBM MQ.

Nastavení vlastností zprávy

Volejte funkci MQSETMP pro každou vlastnost, kterou chcete nastavit. Když vložíte zprávu do pole popisovač zprávy a pole akce struktury MQPMO, nastavte ji.

Chcete-li přidružit vlastnosti ke zprávě, musí mít zpráva popisovač zprávy. Vytvořte popisovač zprávy pomocí volání funkce MQCRTMH. Volejte funkci MQSETMP, která určuje tento popisovač zprávy pro každou vlastnost, kterou chcete nastavit. K dispozici je ukázkový program amqsstma.cpro ilustraci použití příkazu MQSETMP.

Pokud se jedná o novou zprávu, když ji vložíte do fronty pomocí příkazu MQPUT nebo MQPUT1, nastavte pole OriginalMsgv MQPMO na hodnotu tohoto manipulátoru zprávy a nastavte pole akce MQPMO na hodnotu MQACTP_NEW (jedná se o výchozí hodnotu).

Pokud se jedná o zprávu, kterou jste již dříve získali, a vy nyní předáváte nebo odpovídáte na ni nebo na odeslání sestavy v odpovědi, umístěte původní popisovač zprávy do pole OriginalMsgobslužné rutiny MQPMO a nový popisovač zprávy v poli Popisovač NewMsg. Podle potřeby nastavte pole Akce na hodnotu MQACTP_FORWARD, MQACTP_REPLY nebo MQACTP_REPORT.

Máte-li vlastnosti v záhlaví MQRFH2 ze zprávy, kterou jste již dříve získali, můžete je převést na vlastnosti obslužné rutiny zpráv pomocí volání MQBUFMH.

Pokud vystavujete zprávu do fronty ve správci front na úrovni dřívější než IBM WebSphere MQ 7.0, která nemůže zpracovat vlastnosti zprávy, můžete nastavit parametr PropertyControl v definici kanálu tak, aby určoval, jak mají být tyto vlastnosti zpracovávány.

Řízení informací o kontextu zprávy

Použijete-li volání MQPUT nebo MQPUT1 k vložení zprávy do fronty, můžete určit, že správce front má přidat některé výchozí kontextové informace do deskriptoru zpráv. Aplikace, které mají odpovídající úroveň oprávnění, mohou přidávat další informace o kontextu. Pole voleb ve struktuře MQPMO můžete použít k řízení informací o kontextu.

Informace o kontextu zprávy umožňují aplikaci, která načte zprávu, aby zjistila informace o odesílateli zprávy. Všechny kontextové informace jsou uloženy v kontextových polích deskriptoru zpráv. Typ informací spadá do kontextu identity, původu a kontextu uživatele.

Chcete-li řídit informace o kontextu, použijte pole *Options* ve struktuře MQPMO.

Nezadáte-li žádné volby pro kontextové informace, správce front přepíše informace o kontextu, které již mohou být v deskriptoru zpráv, informacemi o identitě a kontextu, které vygenerovala pro vaši zprávu. To je stejné jako uvedení volby MQPMO_DEFAULT_CONTEXT.Tyto výchozí informace o kontextu můžete chtít při vytváření nové zprávy (například při zpracování vstupu uživatele z dotazové obrazovky).

Pokud nechcete k dané zprávě přidružit žádné kontextové informace, použijte volbu MQPMO_NO_CONTEXT. Při vkládání zprávy bez kontextu jsou jakékoli kontroly oprávnění provedené produktem IBM MQ prováděny s použitím prázdného ID uživatele. Prázdné ID uživatele nemůže být přiřazeno explicitnímu oprávnění k prostředkům produktu IBM MQ , ale je zpracováno jako člen speciální skupiny 'nobody'. Další podrobnosti o speciální skupině nobodynaleznete v tématu Referenční informace o rozhraní služeb instalovatelných služeb.

Nastavení kontextu můžete provést pomocí funkce MQOPEN následovanou operací MQPUT s použitím volby MQOO_ a MQPMO_ uvedeným v následujících sekcích. Kontextové nastavení můžete také nastavit pouze pomocí hodnoty MQPUT1, v takovém případě stačí vybrat volbu MQPMO_ uvedenou v níže uvedených sekcích.

Následující části tohoto tématu popisují použití kontextu identity, kontextu uživatele a všech kontextů.

- “Předání kontextu identity” na stránce 730
- “Předání kontextu uživatele” na stránce 730
- “Předávání všech kontextů” na stránce 730
- “Nastavení kontextu identity” na stránce 730
- “Nastavení kontextu uživatele” na stránce 731
- “Nastavení celého kontextu” na stránce 731

Předání kontextu identity

Obecně řečeno, programy by měly předávat informace o kontextu identity ze zprávy do zprávy okolo aplikace, dokud data nedosáhne svého konečného cíle.

Programy by měly při každé změně dat změnit informace o kontextu původu. Aplikace, které chtějí změnit nebo nastavit jakékoli informace o kontextu, však musí mít odpovídající úroveň oprávnění. Správce front zkontroluje toto oprávnění, pokud aplikace otevřou fronty. Musí mít oprávnění k použití příslušných kontextových voleb pro volání MQOPEN.

Pokud vaše aplikace obdrží zprávu, zpracuje data z této zprávy, pak vloží změněná data do jiné zprávy (případně pro zpracování jinou aplikací), aplikace musí předat informace o kontextu identity z původní zprávy do nové zprávy. Můžete povolit správci front vytvořit informace o kontextu původu.

Chcete-li uložit informace o kontextu z původní zprávy, použijte volbu MQOO_SAVE_ALL_CONTEXT, když otevřete frontu pro získání zprávy. To je dodatkem k dalším volbám, které používáte při volání MQOPEN. Všimněte si však, že informace o kontextu nelze uložit, pokud pouze procházíte zprávou.

Při vytvoření druhé zprávy:

- Otevřete frontu s použitím volby MQOO_PASS_IDENTITY_CONTEXT (navíc k volbě MQOO_OUTPUT).
- V poli *Context* struktury voleb vložení zprávy zadejte popisovač fronty, ze které jste uložili informace o kontextu.
- V poli *Options* struktury příkazu put-message určete volbu MQPMO_PASS_IDENTITY_CONTEXT.

Předání kontextu uživatele

Nemůžete se rozhodnout předat pouze kontext uživatele. Chcete-li předat kontext uživatele při vložení zprávy, uveďte MQPMO_PASS_ALL_CONTEXT. Všechny vlastnosti v kontextu uživatele jsou předávány stejným způsobem jako kontext původu.

Je-li provedeno volání MQPUT nebo MQPUT1 a kontext je předáván, všechny vlastnosti v kontextu uživatele jsou předány z načtené zprávy do zprávy vložení. Všechny vlastnosti kontextu uživatele, které vkládá aplikace do aplikace, se umístí s jejich původními hodnotami. Všechny vlastnosti kontextu uživatele, které aplikace uvedení do provozu odstranily, jsou obnoveny ve zprávě vložení. Zadrží se všechny vlastnosti kontextu uživatele, které vkládá aplikace vkládání do zprávy.

Předávání všech kontextů

Pokud vaše aplikace obdrží zprávu a vloží do jiné zprávy data zprávy (nezměněná), aplikace musí předat všechny informace o kontextu (identita, původ a uživatel) z původní zprávy do nové zprávy. Příkladem aplikace, která by mohla být tato, je modul pro přesouvání zpráv, který přesouvá zprávy z jedné fronty do jiné.

Postupujte stejným postupem jako při předávání kontextu identity, kromě toho, že jste použili volbu MQOTE_PASS_ALL_CONTEXT a volbu vložení zprávy MQPMO_PASS_ALL_CONTEXT.

Nastavení kontextu identity

Chcete-li nastavit informace o kontextu identity pro zprávu:

- Otevřete frontu s použitím volby MQOO_SET_IDENTITY_CONTEXT.
- Vložte zprávu do fronty a zadejte volbu MQPMO_SET_IDENTITY_CONTEXT. V deskriptoru zpráv zadejte jakékoli informace o kontextu identity, které požadujete.

Poznámka: Nastavíte-li některé (ale ne všechny) pole kontextu identity pomocí voleb MQOO_SET_IDENTITY_CONTEXT a MQPMO_SET_IDENTITY_CONTEXT, je důležité si uvědomit, že správce front nenastavuje žádná jiná pole.

Chcete-li upravit kteroukoli z voleb kontextu zprávy, musíte mít příslušná oprávnění k vydávání volání. Chcete-li například použít funkci MQOO_SET_IDENTITY_CONTEXT nebo MQPMO_SET_IDENTITY_CONTEXT, musíte mít oprávnění `+setid` .

Nastavení kontextu uživatele

Chcete-li nastavit vlastnost v kontextu uživatele, nastavte pole Kontext deskriptoru vlastnosti zprávy (MQPD) na hodnotu MQPD_USER_CONTEXT při volání funkce MQSETMP.

K nastavení vlastnosti v kontextu uživatele nepotřebujete žádné speciální oprávnění. Kontext uživatele nemá žádné volby kontextu MQOO_SET_ * nebo MQPMO_SET_ *.

Nastavení celého kontextu

Chcete-li nastavit jak identitu, tak informace o kontextu původu pro zprávu, postupujte takto:

1. Otevřete frontu s použitím volby MQOO_SET_ALL_CONTEXT.
2. Vložte zprávu do fronty uvedením volby MQPMO_SET_ALL_CONTEXT. V deskriptoru zpráv zadejte libovolnou informaci o identitě a původu, kterou požadujete.

Pro každý typ nastavení kontextu je zapotřebí příslušné oprávnění.

Související pojmy

"kontext zprávy" na stránce 41
Informace *Kontext zprávy* umožňují aplikaci, která načte zprávu, zjistit informace o odesílateli zprávy.

Související odkazy

"Volby MQOPEN vztahující se ke kontextu zprávy" na stránce 720
Chcete-li mít možnost přidružit kontextové informace ke zprávě, když ji vložíte do fronty, musíte při otevření fronty použít jednu z voleb kontextu zprávy.

Vložení jedné zprávy do fronty pomocí volání MQPUT1

Použijte volání MQPUT1 , chcete-li zavřít frontu okamžitě poté, co jste do ní vložili jednu zprávu. Například serverová aplikace pravděpodobně používá volání MQPUT1 , když odesílá odpověď na každou z různých front.

Hodnota MQPUT1 je funkčně ekvivalentní volání MQOPEN následované operací MQPUT následovaným příkazem MQCLOSE. Jediným rozdílem v syntaxi pro volání MQPUT a MQPUT1 je to, že pro příkaz MQPUT uvedete popisovač objektu, zatímco pro MQPUT1 určujete strukturu deskriptoru objektu (MQOD), jak je definováno v MQOPEN (viz "Označení objektů (struktura MQOD)" na stránce 715). Důvodem je skutečnost, že je třeba předat informace o volání MQPUT1 ke frontě, kterou má otevřít, zatímco při volání funkce MQPUT se musí fronta již otevřít.

Jako vstup do volání MQPUT1 je třeba dodat:

- Popisovač připojení.
- Popis objektu, který chcete otevřít. Tato hodnota je ve formě struktury deskriptoru objektu (MQOD).
- Popis zprávy, kterou chcete vložit do fronty. To je ve formě struktury deskriptoru zpráv (MQMD).
- Řídicí informace ve formě struktury voleb put-message (MQPMO).
- Délka dat obsažených ve zprávě (MQLONG).
- Adresa dat zprávy.

Výstup příkazu MQPUT1 je následující:

- Kód dokončení
- Kód příčiny

Pokud je volání úspěšně dokončeno, vrací také strukturu vaší volby a strukturu deskriptoru zprávy. Volání upravuje strukturu voleb tak, aby zobrazovala název fronty a správce front, do kterého byla zpráva odeslána. Pokud požadujete, aby správce front generoval jedinečnou hodnotu pro identifikátor zprávy, kterou umisťujete (zadáním binární nuly v poli *MsgId* struktury MQMD), zavolání vloží hodnotu do pole *MsgId* před vrácením této struktury zpět.

Poznámka: MQPUT1 nelze použít s názvem modelové fronty, avšak po otevření modelové fronty můžete k dynamické frontě vydat příkaz MQPUT1 .

Je šest vstupních parametrů pro MQPUT1 :

Hconn
Jedná se o manipulátor připojení. Pro aplikace produktu CICS můžete zadat konstantu MQHC_DEF_HCONN (která má hodnotu nula), nebo použít manipulátor připojení vrácený voláním MQCONN nebo MQCONNX. Pro ostatní programy vždy použijte obslužnou rutinu připojení vrácenou voláním MQCONN nebo MQCONNX.

ObjDesc
Jedná se o strukturu deskriptoru objektu (MQOD).

Do polí *ObjectName* a *ObjectQMgrName* zadejte název fronty, do které chcete vložit zprávu, a název správce front, který je vlastníkem této fronty.

Pole *DynamicQName* je pro volání MQPUT1 ignorováno, protože nemůže používat modelové fronty.

Použijte pole *AlternateUserId* , chcete-li navrhnout alternativní identifikátor uživatele, který má být použit pro testovací oprávnění k otevření fronty.

MsgDesc
Jedná se o strukturu deskriptoru zpráv (MQMD). Stejně jako v případě volání MQPUT použijte tuto strukturu k definování zprávy, kterou vkládáte do fronty.

PutMsgOpts
Jedná se o strukturu voleb vložení zprávy (MQPMO). Použijte jej, jak byste měli pro volání MQPUT (viz “Určení voleb pomocí struktury MQPMO” na stránce 725).

Je-li pole *Options* nastaveno na hodnotu nula, správce front používá vlastní ID uživatele při provádění testů pro oprávnění pro přístup k frontě. Správce front bude také ignorovat alternativní identifikátor uživatele zadaný v poli *AlternateUserId* struktury MQOD.

BufferLength
Toto je délka vaší zprávy.

Buffer
Jedná se o oblast vyrovnávací paměti, která obsahuje text vaší zprávy.

Pokud používáte klastry, funkce MQPUT1 bude fungovat, jako by bylo v platnosti vlastnost MQOO_BIND_NOT_FIXED. Aplikace musí používat vyřešená pole ve struktuře MQPMO místo struktury MQOD k určení, kam byla zpráva odeslána. Další informace naleznete v tématu Konfigurace klastru správce front .

Je uveden popis volání MQPUT1 v umístění MQPUT1.

Distribuční seznamy

Nepodporováno na IBM MQ for z/OS. Distribuční seznamy umožňují vložit zprávu do více míst určení v rámci jednoho volání MQPUT nebo MQPUT1 .Jedno volání MQOPEN může otevřít více front a jediné volání MQPUT může poté vložit zprávu do každé z těchto front. Některé generické informace ze struktur MQI použitých pro tento proces mohou být nahrazeny specifickými informacemi týkajícími se jednotlivých míst určení zahrnutých do rozdělovníku.

V 9.0.0.1 V 9.0.1

Upozornění: Distribuční seznamy nepodporují použití alias fronta, které odkazují na objekty tématu. Pokud z IBM MQ 9.0.1 a IBM MQ 9.0.0 Fix Pack 1odkazuje alias fronty na objekt tématu v rozdělovníku, IBM MQ vrátí MQRC_ALIAS_BASE_Q_TYPE_ERROR.

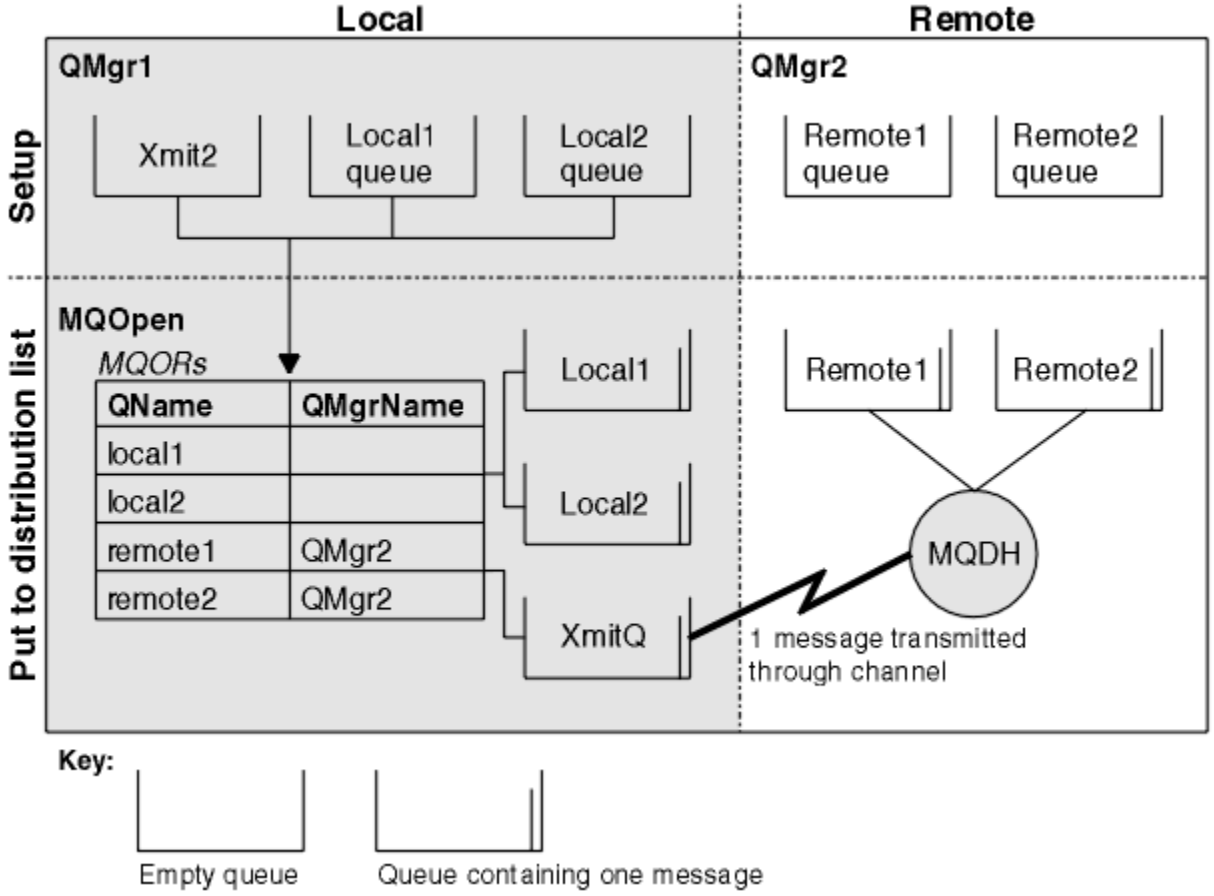
Je-li vydáno volání MQOPEN, jsou obecné informace převzaty z deskriptoru objektu (MQOD). Pokud uvedete MQOD_VERSION_2 do pole *Version* a hodnotu větší než nula v poli *RecsPresent* , *Hobj* může být definováno jako popisovač seznamu (jedné nebo více front), spíše než fronty. V tomto případě jsou specifické informace poskytnuty prostřednictvím záznamů objektů (MQORs), které uvádějí podrobnosti o místě určení (to znamená *ObjectName* a *ObjectQMgrName*).

Manipulátor objektu (*Hobj*) je předán volání MQPUT, což umožňuje vložení do jedné fronty a nikoli do jediné fronty.

Je-li zpráva vložena do fronty (MQPUT), jsou generické informace převzaty ze struktury volby vložení zprávy (MQPMO) a MQMD (Message Descriptor). Specifické informace jsou uvedeny ve formě záznamů vložení zpráv (MQPMRs).

Záznamy odpovědí (MQRR) mohou přijmout kód dokončení a kód příčiny specifický pro každou cílovou frontu.

Příkaz Obrázek 66 na stránce 733 ukazuje, jak distribuční seznamy fungují.



Obrázek 66. Jak distribuční seznamy fungují

Otevírání distribučních seznamů
Použijte volání MQOPEN k otevření rozdělovníku a použijte volby volání k uvedení toho, co chcete dělat se seznamem.

Jako vstup pro volání MQOPEN je třeba zadat:

- Popisovač připojení (viz “Vložení zpráv do fronty” na stránce 723 pro popis)
- Generické informace ve struktuře deskriptoru objektu (MQOD)
- Název každé fronty, kterou chcete otevřít, pomocí struktury záznamů objektů (MQOR)

Výstup z MQOPEN je:

- Popisovač objektu, který představuje váš přístup k rozdělovníku
- Obecný kód dokončení
- Generický kód příčiny
- Záznamy odpovědí (volitelné), obsahující kód dokončení a důvod pro každé místo určení

Použití struktury MQOD

Strukturu MQOD použijte k identifikaci front, které chcete otevřít.

Chcete-li definovat distribuční seznam, musíte zadat MQOD_VERSION_2 do pole *Version* , hodnota větší než nula v poli *RecsPresent* a MQOT_Q v poli *ObjectType* . Popis všech polí struktury MQOD najdete v tématu MQOD .

Použití struktury MQOR

Zadejte strukturu MQOR pro každý cíl.

Struktura obsahuje názvy cílové fronty a správce front. Pole *ObjectName* a *ObjectQMgrName* v MQOD se nepoužívají pro distribuční seznamy. Musí existovat jeden nebo více záznamů objektů. Je-li pole *ObjectQMgrName* ponecháno prázdné, použije se lokální správce front. Další informace o těchto polích viz ObjectName a ObjectQMgrName .

Cílové fronty lze určit dvěma způsoby:

- Použitím pole offsetu *ObjectRecOffset*.

  V takovém případě aplikace musí deklarovat vlastní strukturu obsahující strukturu MQOD, za kterou bude následovat pole záznamů MQOR (s tolika prvky pole, kolik je potřeba), a nastavit proměnnou *ObjectRecOffset* na posun prvního prvku v poli od začátku MQOD. Ujistěte se, že je tento posun správný.

  Doporučuje se použití vestavěných systémových prostředků poskytnutých programovacím jazykem, pokud jsou k dispozici ve všech prostředích, ve kterých je aplikace spuštěna. Následující kód ilustruje tuto techniku pro programovací jazyk COBOL:

```
01  MY-OPEN-DATA.
    02 MY-MQOD.
       COPY CMQODV.
    02 MY-MQOR-TABLE OCCURS 100 TIMES.
       COPY CMQORV.
    MOVE LENGTH OF MY-MQOD TO MQOD-OBJECTRECOFFSET.
```

  Případně můžete použít konstantní MQOD_CURRENT_LENGTH, pokud programovací jazyk nepodporuje nezbytná vestavěná zařízení ve všech dotčených prostředích. Následující kód ilustruje tuto techniku:

```
01  MY-MQ-CONSTANTS.
    COPY CMQV.
01  MY-OPEN-DATA.
    02 MY-MQOD.
       COPY CMQODV.
    02 MY-MQOR-TABLE OCCURS 100 TIMES.
       COPY CMQORV.
    MOVE MQOD-CURRENT-LENGTH TO MQOD-OBJECTRECOFFSET.
```

  To však funguje správně pouze v případě, že struktura MQOD a pole záznamů MQOR jsou souvislé; pokud kompilátor vkládá mezi MQOD a MQOR pole přeskočených bajtů, je nutné je přidat k hodnotě uložené v produktu *ObjectRecOffset*.

  Použití *ObjectRecOffset* se doporučuje pro programovací jazyky, které nepodporují datový typ ukazatele, nebo které implementují datový typ ukazatele takovým způsobem, který není přenosný do různých prostředí (například programovací jazyk COBOL).

- Použitím pole ukazatele *ObjectRecPtr*.

  V takovém případě může aplikace deklarovat pole struktury MQOR odděleně od struktury MQOD a nastavit *ObjectRecPtr* na adresu pole. Následující kód ilustruje tuto techniku pro programovací jazyk C:

```
MQOD MyMqod;
MQOR MyMqor[100];
MyMqod.ObjectRecPtr = MyMqor;
```

  Použití *ObjectRecPtr* se doporučuje pro programovací jazyky, které podporují datový typ ukazatele způsobem, který je přenosný do různých prostředí (například programovací jazyk C).

Zvolená technika, kterou si zvolíte, musíte použít jeden z produktů *ObjectRecOffset* a *ObjectRecPtr* ; volání selže s kódem příčiny MQRC_OBJECT_RECORDS_ERROR, pokud jsou obě hodnoty nula nebo obě jsou nenulové.

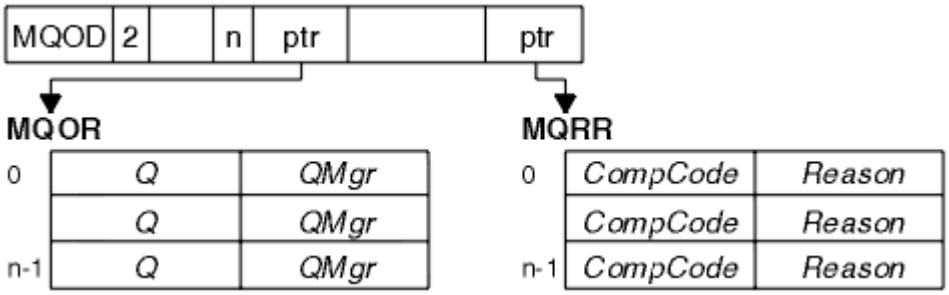
Použití struktury MQRR

Tyto struktury jsou specifické pro cíl; každý záznam odpovědi obsahuje pole *CompCode* a *Reason* pro každou frontu distribučního seznamu. Tuto strukturu musíte použít k tomu, abyste mohli rozlišovat, kde se vyskytují nějaké problémy.

Pokud například obdržíte kód příčiny MQRC_MULTIPLE_REASONS a váš distribuční seznam obsahuje pět cílových front, nebudete vědět, do kterých front se tyto problémy vztahují, pokud tuto strukturu nepoužijete. Pokud však máte kód dokončení a kód příčiny pro každé místo určení, můžete chyby snadněji lokalizovat.

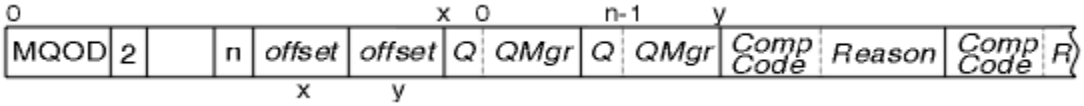
Další informace o struktuře MQRR najdete v tématu MQRR .

Obrázek 67 na stránce 735 ukazuje, jak můžete otevřít distribuční seznam v C.



Obrázek 67. Otevření distribučního seznamu v jazyce C

Produkt Obrázek 68 na stránce 735 zobrazuje, jak můžete otevřít distribuční seznam v jazyce COBOL.



Obrázek 68. Otevření distribučního seznamu v jazyce COBOL

Použití voleb MQOPEN

Při otevírání distribučního seznamu můžete určit následující volby:

- MQOOK_VÝSTUP
- MQOO_FAIL_IF_QUIESCING (volitelné)
- MQONE_ALTERNATE_USER_AUTHORITY (volitelné)
- MQOO_ * _CONTEXT (volitelné)

Popis těchto voleb najdete v části “Otevírání a zavírání objektů” na stránce 712 .

Vložení zpráv do distribučního seznamu
Chcete-li vložit zprávy do distribučního seznamu, můžete použít příkaz MQPUT nebo MQPUT1.

Jako vstup musíte dodat:

- Popisovač připojení (viz “Vložení zpráv do fronty” na stránce 723 pro popis).
- Popisovač objektu. Je-li distribuční seznam otevřen pomocí funkce MQOPEN, produkt *Hobj* umožňuje pouze vložení do seznamu.
- Struktura deskriptoru zpráv (MQMD). Popis této struktury viz MQMD .

- Řídicí informace v podobě struktury volby put-message (MQPMO). Chcete-li získat informace o vyplnění polí struktury MQPMO, prohlédněte si příručku “Určení voleb pomocí struktury MQPMO” na stránce 725 .
- Řídicí informace ve formátu záznamů vložení zpráv (MQPMR).
- Délka dat obsažených ve zprávě (MQLONG).
- Samotná data zprávy.

Výstup je:

- Kód dokončení
- Kód příčiny
- Záznamy odpovědí (volitelné)

Použití struktury MQPMR

Tato struktura je volitelná a uvádí informace specifické pro místo určení pro některá pole, která byste mohli chtít identifikovat jinak než hodnoty, které jsou již identifikovány v MQMD.

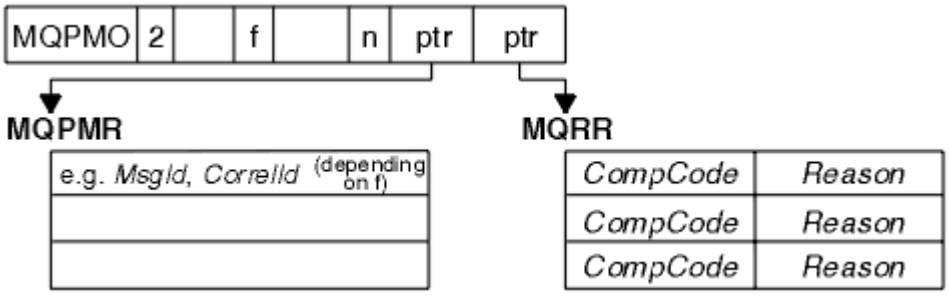
Popis těchto polí naleznete v tématu MQPMR.

Obsah jednotlivých záznamů závisí na informacích uvedených v poli *PutMsgRecFields* MQPMO. Příklad: v ukázkovém programu AMQSPTL0.C (viz “Ukázkový program Distribuční seznam” na stránce 1065 pro popis) zobrazující použití distribučních seznamů, ukázka zvolí zadání hodnot pro *MsgId* a *CorrelId* v MQPMR. Tato část ukázkového programu vypadá takto:

```
typedef struct
{
MQBYTE24 MsgId;
MQBYTE24 CorrelId;
} PutMsgRec;
...
/**********************
MQLONG PutMsgRecFields=MQPMRF_MSG_ID | MQPMRF_CORREL_ID;
```

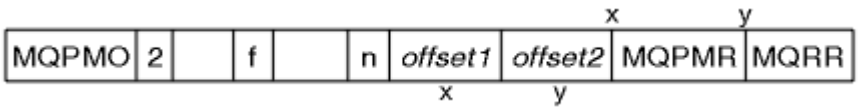
To znamená, že *MsgId* a *CorrelId* jsou poskytnuty pro každé místo určení rozdělovníku. Záznamy Put Message Records jsou poskytovány jako pole.

Obrázek 69 na stránce 736 ukazuje, jak můžete vložit zprávu do rozdělovníku v C.



Obrázek 69. Vložení zprávy do distribučního seznamu v jazyce C

Produkt Obrázek 70 na stránce 736 zobrazuje, jak můžete vložit zprávu do distribučního seznamu v jazyce COBOL.



Obrázek 70. Vložení zprávy do distribučního seznamu v jazyce COBOL

Použití příkazu MQPUT1

Používáte-li MQPUT1, zvažte následující body:

1. Hodnoty polí *ResponseRecOffset* a *ResponseRecPtr* musí být null nebo nula.
2. Záznamy odpovědí, je-li to nutné, musí být adresovány z MQOD.

Některé případy, kdy volání vložení selžou

Změní-li se některé atributy fronty použitím příkazu FORCE u příkazu během intervalu mezi zadáním volání MQOPEN a voláním MQPUT, volání MQPUT selže a vrátí kód příčiny MQRC_OBJECT_CHANGED.

Správce front označí obslužnou rutinu objektu jako neplatnou. K tomu dojde také v případě, že dojde ke zpracování změn během zpracování volání MQPUT1 nebo v případě, že se změny použijí pro každou frontu, na kterou je název fronty vyřešen. Atributy, které ovlivňují popisovač tímto způsobem, jsou uvedeny v popisu volání MQOPEN v MQOPEN. Pokud vaše volání vrátí kód příčiny MQRC_OBJECT_CHANGED, zavřete frontu a znovu ji otevřete a poté zkuste zprávu vložit znovu.

Jsou-li operace vložení zakázány pro frontu, na kterou se pokoušíte vložit zprávy (nebo frontu, na kterou se název fronty rozlišuje), volání MQPUT nebo MQPUT1 selže a vrátí kód příčiny MQRC_PUT_INHIBITED. Pokud se později pokusíte o telefonát, může být zpráva úspěšně vložena, je-li návrh aplikace takový, že jiné programy pravidelně mění atributy front.

Je-li fronta, do které se pokoušíte vložit zprávu, zaplněna, volání MQPUT nebo MQPUT1 selže a vrátí MQRC_Q_FULL.

Byla-li odstraněna dynamická fronta (dočasná nebo trvalá), volání MQPUT s použitím dříve získaného manipulátoru objektu selže a vrátí kód příčiny MQRC_Q_DELETED. V této situaci je dobrým zvykem zavřít popisovač objektu, protože již pro vás není žádný jiný způsob použití.

V případě distribučních seznamů se může v jednom požadavku vyskytnout více kódů dokončení a kódů příčiny. Nelze je zpracovat pouze pomocí výstupních polí *CompCode* a *Reason* v MQOPEN a MQPUT.

Použijete-li distribuční seznamy k umístění zpráv do více míst určení, obsahují záznamy odpovědí specifické *CompCode* a *Reason* pro každý cíl. Obdržíte-li kód dokončení MQCC_FAILED, žádná zpráva nebyla úspěšně vložena do žádné cílové fronty. Je-li kód dokončení MQCC_WARNING, zpráva se úspěšně umístí do jedné nebo více cílových front. Pokud obdržíte návratový kód MQRC_MULTIPLE_REASONS, nejsou všechny kódy příčiny všechny stejné pro všechny cíle. Proto je doporučeno použít strukturu MQRR tak, abyste mohli určit, které fronty nebo fronty způsobily chybu a důvody pro každou z nich.

Získávání zpráv z fronty

Tyto informace použijte k získání informací o získávání zpráv z fronty.

Zprávy z fronty lze získat dvěma způsoby:

1. Můžete odebrat zprávu z fronty, aby ji již ostatní programy nevidět.
2. Můžete zkopírovat zprávu a ponechat původní zprávu ve frontě. To se označuje jako *procházení*. Zprávu můžete odebrat, jakmile ji prohlédnou.

V obou případech je třeba použít volání MQGET, ale nejprve musí být aplikace připojena ke správci front a musíte použít volání MQOPEN k otevření fronty (pro vstup, procházení nebo obojí). Tyto operace jsou popsány v “Připojování k správci front a odpojování od něj” na stránce 704 a “Otevírání a zavírání objektů” na stránce 712.

Po otevření fronty můžete opakovaně používat volání MQGET k procházení nebo odebírání zpráv ve stejné frontě. Po dokončení získávání všech zpráv, které chcete z fronty, zavolejte funkci MQCLOSE.

Chcete-li zjistit více o získávání zpráv z fronty, použijte následující odkazy:

- “Získávání zpráv z fronty pomocí volání MQGET” na stránce 738
- “Pořadí, ve kterém jsou zprávy načítány z fronty” na stránce 742
- “Získání konkrétní zprávy” na stránce 753
- “Zlepšení výkonu přechodných zpráv” na stránce 754

- z/OS "Typ indexu" na stránce 758
- "Zpracování zpráv větších než 4 MB" na stránce 759
- "Čekání na zprávy" na stránce 764
- z/OS "signalizace" na stránce 764
- "Vynechání odvolání" na stránce 766
- "Převod dat aplikace" na stránce 768
- "Procházení zpráv ve frontě" na stránce 769
- "Některé případy, kdy se volání MQGET nezdaří" na stránce 774

Související pojmy

"Přehled rozhraní fronty zpráv" na stránce 690
Zde jsou uvedeny informace o komponentách rozhraní MQI (Message Queue Interface).

"Připojování k správci front a odpojování od něj" na stránce 704
Chcete-li používat programovací služby IBM MQ , musí mít program připojení ke správci front. V této části se dozvíte, jak se připojit k správci front a odpojit je od správce front.

"Otevírání a zavírání objektů" na stránce 712
Tyto informace poskytují vhled do otevírání a zavírání objektů IBM MQ .

"Vložení zpráv do fronty" na stránce 723
V této části se dozvíte, jak vložit zprávy do fronty.

"Inquaning about a nastavení atributů objektu" na stránce 815
Atributy jsou vlastnosti, které definují charakteristiky objektu IBM MQ .

"Potvrzení a zálohování jednotek práce" na stránce 818
Tyto informace popisují, jak potvrdit a zazálohovat všechny zotavitelné operace get a put, které se vyskytly v jednotce práce.

"Spuštění aplikací produktu IBM MQ pomocí spouštěčů" na stránce 829
Informace o spouštěčích a o tom, jak spustit aplikace IBM MQ pomocí spouštěčů.

"Práce s MQI a klastry" na stránce 847
Existují speciální volby na voláních a návratových kódech, které se vztahují ke klastrování.

"Použití a zápis aplikací v systému IBM MQ for z/OS" na stránce 851
Aplikace produktu IBM MQ for z/OS mohou být vytvořeny z programů spuštěných v mnoha různých prostředích. To znamená, že mohou využít výhody zařízení dostupných ve více než jednom prostředí.

"Aplikace mostu IMS a IMS v systému IBM MQ for z/OS" na stránce 57
Tyto informace vám pomohou při psaní aplikací produktu IMS pomocí produktu IBM MQ.

Získávání zpráv z fronty pomocí volání MQGET

Volání MQGET získá zprávu z otevřené lokální fronty. Nelze získat zprávu z fronty na jiném systému.

Jako vstup pro volání MQGET je třeba dodat:

- Popisovač připojení.
- Popisovač fronty.
- Popis zprávy, kterou chcete získat z fronty. Tato struktura je ve formě struktury deskriptoru zpráv (MQMD).
- Řízení informací ve formě struktury MQGMO (Get Message Options).
- Velikost vyrovnávací paměti, kterou jste přiřadili k zadržení zprávy (MQLONG).
- Adresa úložiště, do kterého se má vložit zpráva.

Výstup z MQGET je:

- Kód příčiny
- Kód dokončení

- Zpráva ve vámi zadané oblasti vyrovnávací paměti, pokud je volání úspěšně dokončeno.
- Vaše struktura voleb, upravená tak, aby zobrazovala název fronty, ze které byla zpráva načtena.
- Struktura deskriptoru zpráv s obsahem polí upravených tak, aby popisovala zprávu, která byla načtena.
- Délka zprávy (MQLONG)

V MQGETje uveden popis volání MQGET.

Následující oddíly popisují informace, které musíte dodat jako vstup pro volání MQGET.

- “Určení manipulátorů připojení” na stránce 739
- “Popisování zpráv pomocí struktury MQMD a volání MQGET” na stránce 739
- “Určení voleb MQGET s použitím struktury MQGMO” na stránce 739
- “Určení velikosti oblasti vyrovnávací paměti” na stránce 741

Určení manipulátorů připojení

z/OS Pro produkt CICS v aplikacích produktu z/OS můžete zadat konstantu MQHC_DEF_HCONN (která má hodnotu nula), nebo použít manipulátor připojení vrácený voláním MQCONN nebo MQCONNX. Pro ostatní aplikace vždy použijte obslužnou rutinu připojení vrácenou voláním MQCONN nebo MQCONNX.

Použijte popisovač fronty (*Hobj*), který je vrácen při volání operace MQOPEN.

Popisování zpráv pomocí struktury MQMD a volání MQGET

Chcete-li identifikovat zprávu, kterou chcete získat z fronty, použijte strukturu deskriptoru zpráv (MQMD).

Jedná se o vstupní/výstupní parametr pro volání MQGET. Existuje úvod do vlastností zprávy, které MQMD popisuje v produktu “Zprávy produktu IBM MQ” na stránce 13, a v MQMDje popis struktury samotné.

Pokud víte, kterou zprávu chcete získat z fronty, prohlédněte si téma “Získání konkrétní zprávy” na stránce 753.

Pokud do konkrétní zprávy nezadáte žádnou zprávu, příkaz MQGET načte ve frontě zprávu *first* . “Pořadí, ve kterém jsou zprávy načítány z fronty” na stránce 742 popisuje, jak je priorita zprávy, atribut **MsgDeliverySequence** fronty a volba MQGMO_LOGICAL_ORDER určují pořadí zpráv ve frontě.

Poznámka: Chcete-li použít příkaz MQGET více než jednou (například při procházení zpráv ve frontě), je třeba po každém volání nastavit pole *MsgId* a *CorrelId* této struktury na hodnotu null. Tato akce vymaže tato pole identifikátorů zprávy, která byla načtena.

Pokud však chcete seskupit zprávy, *GroupId* musí být stejné pro zprávy ve stejné skupině, takže volání vypadá pro zprávu mající stejné identifikátory jako předchozí zpráva, aby se celá skupina mohla vytvořit.

Určení voleb MQGET s použitím struktury MQGMO

Struktura MQGMO je vstupní/výstupní proměnnou pro předání voleb do volání MQGET. Následující sekce vám pomohou dokončit některá pole této struktury.

K dispozici je popis struktury MQGMO v produktu MQGMO.

StrucId
StrucId je čtyřznakové pole použité k identifikaci struktury jako struktury voleb pro získání zprávy. Vždy zadejte MQGMO_STRUC_ID.

Version
*Version*popisuje číslo verze struktury. MQGMO_VERSION_1 je výchozí hodnota. Chcete-li použít pole verze 2 nebo načítat zprávy v logickém pořadí, zadejte MQGMO_VERSION_2. Chcete-li použít pole verze 3 nebo načítat zprávy v logickém pořadí, zadejte MQGMO_VERSION_3. Funkce MQGMO_CURRENT_VERSION nastaví aplikaci tak, aby používala nejnovější úroveň.

Options

V rámci vašeho kódu můžete vybrat volby v libovolném pořadí; každá volba je v poli *Options* reprezentována trochou.

Ovládací prvky pole *Options* :

- Určuje, zda volání MQGET čeká na příchod zprávy do fronty před jejím dokončením (viz “Čekání na zprávy” na stránce 764).
- Zda je operace získání zahrnuta do pracovní jednotky.
- Určuje, zda je trvalá zpráva načtena mimo synchronizační bod, což umožňuje rychlé zasílání zpráv.
- z/OS V systému IBM MQ for z/OS, zda je načtená zpráva označena jako přeskočení odvolání (viz “Vynechání odvolání” na stránce 766)
- Určuje, zda je zpráva odebrána z fronty, nebo pouze procházená
- Zda se má vybrat zpráva pomocí kurzoru pro procházení nebo jiných kritérií výběru
- Zda je volání úspěšné i v případě, že zpráva je delší než vaše vyrovnávací paměť
- z/OS V systému IBM MQ for z/OS, zda povolit dokončení volání. Tato volba také nastaví signál informující o tom, že chcete být upozorněni při doručení zprávy
- Zda se volání nezdaří, je-li správce front ve stavu uvedení do klidového stavu
- z/OS V případě produktu IBM MQ for z/OSbez ohledu na to, zda se volání nezdaří, je-li připojení ve stavu uvedení do klidového stavu
- Zda se vyžaduje konverze dat zpráv aplikace (viz “Převod dat aplikace” na stránce 768)
- Pořadí, ve kterém jsou zprávy a segmenty načteny z fronty z/OS (kromě IBM MQ for z/OS)
- Ať už úplné, logické zprávy lze načítat pouze z/OS (kromě IBM MQ for z/OS)
- Zda se zprávy ve skupině mohou načítat pouze tehdy, když jsou k dispozici *všechny* zprávy ve skupině
- Zda se segmenty v logické zprávě mohou načítat pouze tehdy, jsou-li k dispozici *všechny* segmenty v logické zprávě, z/OS (kromě IBM MQ for z/OS)

Pokud ponecháte pole *Options* nastaveno na výchozí hodnotu (MQGMO_NO_WAIT), volání MQGET funguje tímto způsobem:

- Pokud neexistuje žádná zpráva odpovídající kritériím výběru ve frontě, volání nebude čekat na příchod zprávy, ale provede se okamžitě. z/OS V produktu IBM MQ for z/OStaké volání nenastavuje signál požadující upozornění, když přijde taková zpráva.
- Způsob, jakým volání pracuje se synchronizačních bodů, je určen platformou:

Platforma	**Pod ovládacím prvkem synchronizačního bodu**
IBM i	Ne
Systémy UNIX and Linux	Ne
z/OS z/OS z/OS	Ano
Systémy Windows	Ne

- z/OS V systému IBM MQ for z/OSnení načtená zpráva označena jako přeskočení odvolání.
- Vybraná zpráva bude odebrána z fronty (není procházena).
- Nepožaduje se žádná konverze dat zprávy aplikace.
- Volání selže, pokud je zpráva delší než vaše vyrovnávací paměť.

WaitInterval
Pole *WaitInterval* uvádí maximální dobu (v milisekundách), po kterou volání MQGET čeká na příchod zprávy do fronty, když použijete volbu MQGMO_WAIT. Pokud do doby uvedené v souboru *WaitInterval* nepřijde žádná zpráva, volání se dokončí a vrátí kód příčiny, který ukazuje, že nebyla nalezena žádná zpráva, která by odpovídala kritériím výběru ve frontě.

z/OS Pokud v produktu IBM MQ for z/OSpoužijete volbu MQGMO_SET_SIGNAL, určuje pole *WaitInterval* čas, pro který je nastaven signál.

Další informace o těchto volbách viz "Čekání na zprávy" na stránce 764 z/OS a "signalizace" na stránce 764 .

Signal1

Produkt Signal1 je podporován pouze v produktu z/OS IBM MQ for z/OS.

Pokud použijete volbu MQGMO_SET_SIGNAL k požadavku, aby byla vaše aplikace oznámena při doručení vhodné zprávy, určete typ signálu v poli *Signal1* . V produktu IBM MQ na všech ostatních platformách je pole *Signal1* vyhrazené a jeho hodnota není významná.

z/OS Další informace viz "signalizace" na stránce 764.

Signal2
Pole *Signal2* je vyhrazeno na všech platformách a jeho hodnota není významná.

z/OS Další informace viz "signalizace" na stránce 764.

ResolvedQName
ResolvedQName je výstupní pole, v němž správce front vrací název fronty (po vyřešení jakéhokoli aliasu), ze kterého byla zpráva načtena.

MatchOptions
Produkt *MatchOptions* řídí kritéria výběru pro příkaz MQGET.

GroupStatus
GroupStatus udává, zda je zpráva, kterou jste načetli, ve skupině.

SegmentStatus
SegmentStatus udává, zda položka, kterou jste načetli, je segmentem logické zprávy.

Segmentation
Segmentation udává, zda je segmentace povolena pro načtenou zprávu.

MsgToken

MsgToken Jedinečně identifikuje zprávu.

ReturnedLength
ReturnedLength je výstupní pole, v němž správce front vrací délku vrácených dat zpráv (v bajtech).

MsgHandle
Manipulátor se zprávou, která má být naplněna vlastnostmi zprávy načítané z fronty. Popisovač byl dříve vytvořen voláním MQCRTMH. Všechny vlastnosti, které jsou již přidruženy k popisovači, jsou před načtením zprávy vymazány.

Určení velikosti oblasti vyrovnávací paměti

V argumentu **BufferLength** příkazu MQGET zadejte velikost oblasti vyrovnávací paměti, která má obsahovat data zpráv, která načítáte. Vy rozhodujete, jak velká by měla být ve třech směrech:

1. Možná již víte, jakou délku zpráv očekáváte od tohoto programu. Je-li tomu tak, uveďte vyrovnávací paměť této velikosti.

   Můžete však použít volbu MQGMO_ACCEPT_TRUNCATED_MSG ve struktuře MQGMO, pokud má být volání MQGET dokončeno i v případě, že je zpráva příliš velká pro vyrovnávací paměť. V tomto případě:

- Vyrovnávací paměť je zaplněna jako velká část zprávy, jak ji lze zadržet.
- Volání vrátí kód dokončení varování.
- Zpráva se odstraní z fronty (zruší zbývající část zprávy) nebo je kurzor procházení zálohován (pokud procházíte frontu)
- Skutečná délka zprávy je vrácena v produktu *DataLength*

Bez této volby je volání stále dokončeno s varováním, ale neodebere zprávu z fronty (nebo zálohuje kurzor procházení).

2. Odhadněte velikost vyrovnávací paměti (nebo dokonce určete velikost nula bajtů) a *nepoužívat* volbu MQGMO_ACCEPT_TRUNCATED_MSG. Pokud se volání MQGET nezdaří (například proto, že vyrovnávací paměť je příliš malá), je délka zprávy vrácena v parametru **DataLength** volání. (Vyrovnávací paměť je stále zaplněna jako velká část zprávy, jak ji lze zadržet, ale zpracování volání není dokončeno.) Uložte *MsgId* této zprávy, pak zopakujte volání MQGET, uvedení oblasti vyrovnávací paměti o správné velikosti a *MsgId* , které jste si poznamenali z prvního volání.

   Pokud váš program obsluhuje frontu, která je také obsluhována jinými programy, může jeden z těchto jiných programů odstranit zprávu, kterou chcete, než bude moci váš program vydat další volání MQGET. Váš program může ztrácet čas hledáním zprávy, která již neexistuje. Chcete-li se tomu vyhnout, nejprve projděte frontu, dokud nenaleznete požadovanou zprávu, zadáním *BufferLength* hodnoty nula a pomocí volby MQGMO_ACCEPT_TRUNCATED_MSG. Tato pozice umístí kurzor pod zprávu, kterou chcete. Poté můžete znovu načíst zprávu vyvoláním příkazu MQGET a určením volby MQGMO_MSG_UNDER_CURSOR. Pokud jiný program odstraní zprávu mezi voláními pro procházení a odebrání, váš druhý příkaz MQGET selže ihned (bez prohledání celé fronty), protože pod vaším kurzorem procházení není žádná zpráva.

3. Atribut *MaxMsgLength queue* určuje maximální délku zpráv přijatých pro tuto frontu; atribut *MaxMsgLength správce front* určuje maximální délku zpráv přijatých pro daného správce front. Pokud nevíte, jakou délku zprávy očekávat, můžete se dotázat na atribut **MaxMsgLength** (pomocí volání MQINQ), pak zadat vyrovnávací paměť této velikosti.

   Pokuste se velikost vyrovnávací paměti co nejblíže ke skutečné velikosti zprávy, aby nedošlo ke snížení výkonu.

   Další informace o atributu **MaxMsgLength** viz "Zvýšení maximální délky zprávy" na stránce 759.

Pořadí, ve kterém jsou zprávy načítány z fronty

Pořadí, ve kterém načítáte zprávy z fronty, můžete řídit. Tato sekce se zaměřuje na volby.

Priorita

Program může při vložení zprávy do fronty přiřadit prioritu zprávy (viz "Priority zpráv" na stránce 21). Zprávy se stejnou prioritou jsou uloženy ve frontě v pořadí příchodu, nikoli pořadí, v jakém jsou potvrzeny.

Správce front udržuje fronty buď ve striktní posloupnosti FIFO (první dovnitř, první ven), nebo ve FIFO v rámci posloupnosti priority. Závisí to na nastavení atributu **MsgDeliverySequence** fronty. Když zpráva dorazí do fronty, je vložena okamžitě za poslední zprávou, která má stejnou prioritu.

Programy mohou buď získat první zprávu z fronty, nebo mohou získat určitou zprávu z fronty a ignorovat prioritu těchto zpráv. Program může například chtít zpracovat odpověď na konkrétní zprávu, kterou odeslal dříve. Další informace viz "Získání konkrétní zprávy" na stránce 753.

Pokud aplikace vkládá posloupnost zpráv do fronty, může další aplikace tyto zprávy načíst ve stejném pořadí, ve kterém byly vloženy, za předpokladu, že:

- Všechny zprávy mají stejnou prioritu
- Všechny zprávy byly vloženy do stejné jednotky práce nebo byly všechny vloženy mimo jednotku práce.
- Fronta je lokální vzhledem k aplikaci vkládání

Pokud tyto podmínky nejsou splněny a aplikace závisí na zprávách, které jsou načítány v určitém pořadí, aplikace musí buď zahrnout informace o posloupnosti do dat zprávy, nebo vytvořit prostředek pro potvrzení přijetí zprávy před odesláním další zprávy.

z/OS V systému IBM MQ for z/OSmůžete použít atribut fronty *IndexType*, abyste zvýšili rychlost operací MQGET ve frontě. Další informace viz “Typ indexu” na stránce 758.

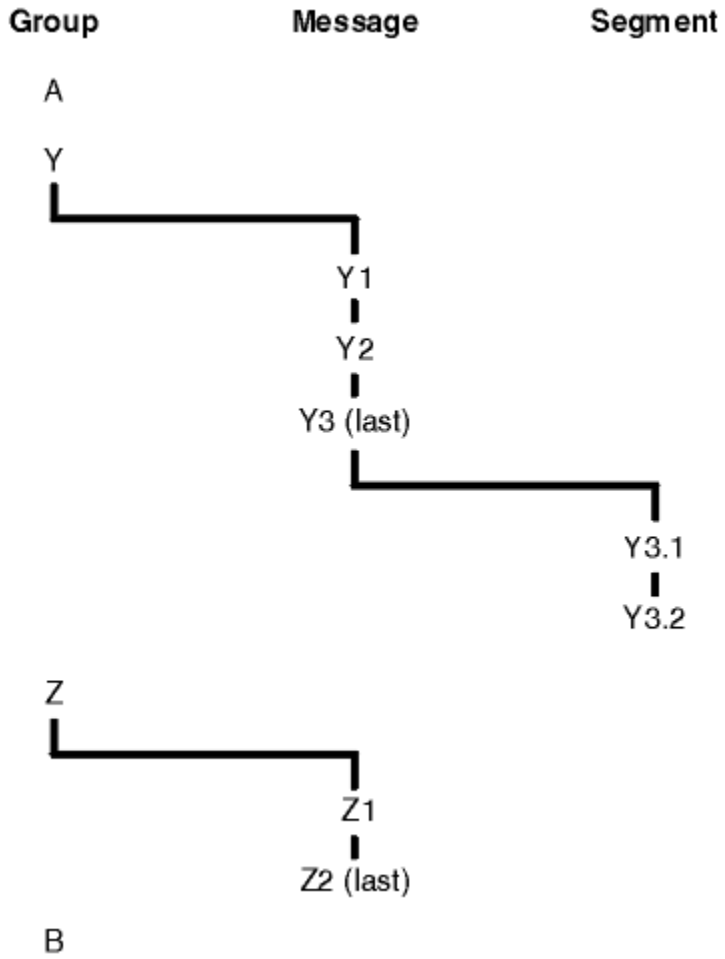
Logické a fyzické uspořádání
Zprávy ve frontách se mohou vyskytnout (v rámci každé úrovně priority) ve *fyzickém* nebo *logickém* pořadí.

Fyzické pořadí je pořadí, ve kterém zprávy dorazí do fronty. Logické pořadí je, když všechny zprávy a segmenty uvnitř skupiny jsou ve své logické posloupnosti, vedle sebe, v pozici určené fyzickou pozicí první položky patřící do skupiny.

Popis skupin, zpráv a segmentů najdete v tématu “Skupiny zpráv” na stránce 38. Tyto fyzické a logické pořadí se mohou lišit, protože:

- Skupiny mohou přijít na místo určení v podobných časech z různých aplikací, takže ztratí jakékoli odlišné fyzické pořadí.
- I v rámci jedné skupiny se mohou zprávy dostat mimo pořadí, protože došlo k přesměrování nebo zpoždění některých zpráv ve skupině.

Například, logický příkaz může vypadat jako obrázek Obrázek 71 na stránce 743:



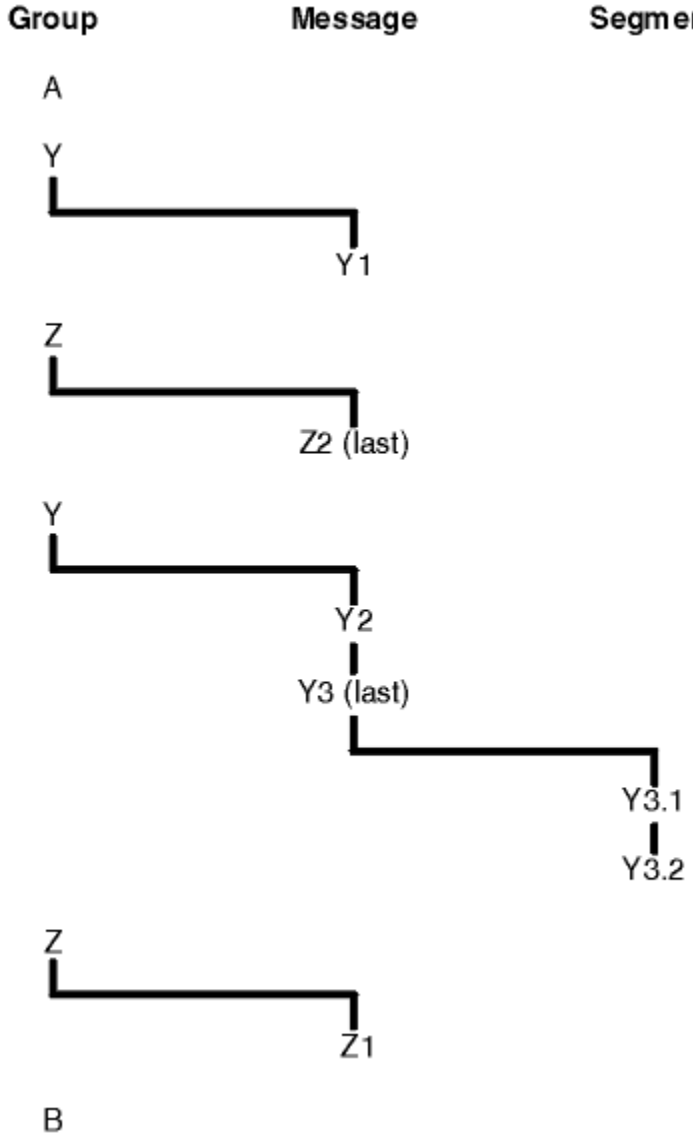
Obrázek 71. Logické pořadí ve frontě

Tyto zprávy se objeví v následujícím logickém pořadí ve frontě:

1. Zpráva A (ne ve skupině)
2. Logická zpráva 1 skupiny Y
3. Logická zpráva 2 skupiny Y
4. Segment 1 z (poslední) logické zprávy 3 skupiny Y

5. (Poslední) segment 2 z (poslední) logické zprávy 3 skupiny Y
6. Logická zpráva 1 skupiny Z
7. (Poslední) logická zpráva 2 skupiny Z
8. Zpráva B (ne ve skupině)

Fyzické uspořádání by však mohlo být zcela odlišné. Fyzická pozice *první* položky v každé skupině určuje logickou pozici celé skupiny. Například, pokud skupiny Y a Z přišly v podobném čase a zpráva 2 skupiny Z převzala zprávu 1 stejné skupiny, fyzické pořadí by vypadalo jako obrázek Obrázek 72 na stránce 744:



Obrázek 72. Fyzické pořadí ve frontě

Tyto zprávy se objevují v následujícím fyzickém pořadí na frontě:

1. Zpráva A (ne ve skupině)
2. Logická zpráva 1 skupiny Y
3. Logická zpráva 2 skupiny Z
4. Logická zpráva 2 skupiny Y
5. Segment 1 z (poslední) logické zprávy 3 skupiny Y
6. (Poslední) segment 2 z (poslední) logické zprávy 3 skupiny Y
7. Logická zpráva 1 skupiny Z

8. Zpráva B (ne ve skupině)

Poznámka: V systému IBM MQ for z/OSnení fyzické pořadí zpráv ve frontě zaručeno, je-li fronta indexována pomocí GROUPID.

Při získávání zpráv můžete zadat MQGMO_LOGICAL_ORDER, chcete-li načítat zprávy v logickém pořadí spíše než ve fyzickém pořadí.

Zadáte-li volání MQGET s klauzulí MQGMO_BRON FIRST a MQGMO_LOGICAL_ORDER, musí následné volání MQGET s MQGMO_BROD NEXT také určovat MQGMO_LOGICAL_ORDER. Naopak, pokud MQGET s MQGMO_BROE_FIRST neurčuje MQGMO_LOGICAL_ORDER, nesmí ani následující MQGET s MQGMO_BRONEM NEXT.

Informace o skupinách a segmentech, které správce front uchovává pro volání MQGET, která procházejí zprávy ve frontě, je oddělena od informací o skupině a segmentu, které správce front uchovává pro volání MQGET, která odebírá zprávy z fronty. Když uvedete MQGMO_BE_FIRST, správce front ignoruje informace o skupině a segmentu pro procházení a prohledá frontu, jako by neexistovala žádná aktuální skupina a žádná aktuální logická zpráva.

Poznámka: Nepoužívejte volání MQGET k procházení *za koncem* skupiny zpráv (nebo logické zprávy, která se nenachází ve skupině) bez určení hodnoty MQGMO_LOGICAL_ORDER. Například, pokud poslední zpráva ve skupině *předchází* první zprávě ve skupině ve frontě, pomocí MQGMO_BERE NEXT k procházení za koncem skupiny, uvedení MQMO_MATCH_MSG_SEQ_NUMBER s `MsgSeqNumber` nastaveným na 1 (vyhledat první zprávu další skupiny) vrátí znovu první zprávu ve skupině již prohlédnuto. K tomu může dojít okamžitě nebo k několika dalším voláním MQGET (pokud jsou mezi nimi nějaké vedlejší skupiny).

Zamezte možnosti nekonečné smyčky otevřením fronty *dvakrát* pro procházení:

- Použijte první popisovač k procházení pouze první zprávy v každé skupině.
- Druhý ovladač použijte k procházení pouze zpráv v rámci určité skupiny.
- Použijte volby MQMO_ * k přesunutí druhého kurzoru pro procházení na pozici prvního kurzoru pro procházení, a teprve pak můžete procházet zprávy ve skupině.
- Nepoužívejte volbu MQGMO_BROWSE NEXT, než je konec skupiny.

Další informace naleznete v tématu MQGET, MQMDa Pravidla pro ověřování platnosti voleb MQI.

U většiny aplikací si při procházení pravděpodobně vyberete logické nebo fyzické uspořádání. Pokud však chcete přepínat mezi těmito režimy, pamatujte na to, že když nejprve zadáte příkaz k procházení MQGMO_LOGICAL_ORDER, bude vaše pozice v rámci logické posloupnosti vytvořena.

Pokud první položka v rámci skupiny není momentálně přítomna, skupina, kterou jste v této době, není považována za součást logické posloupnosti.

Jakmile se kurzor procházení nachází uvnitř skupiny, může pokračovat ve stejné skupině, i když je první zpráva odebrána. Na počátku se však nikdy nemůžete přesunout do skupiny pomocí MQGMO_LOGICAL_ORDER, kde první položka není přítomna.

MQPMO_LOGICAL_ORDER

Volba MQPMO sděluje správci front, jak aplikace vkládá zprávy do skupin a segmentů logických zpráv. Může být zadán pouze na volání MQPUT; není platný na volání MQPUT1 .

Je-li zadán parametr MQPMO_LOGICAL_ORDER, znamená to, že aplikace bude používat následná volání MQPUT, aby:

1. Vložila segmenty do každé logické zprávy kvůli zvýšení offsetu segmentu, počínaje 0, bez mezer.
2. Vložila všechny segmenty do jedné logické zprávy, a teprve pak vložila segment do další logické zprávy.
3. Vložila logické zprávy do každé skupiny zprávy, aby zvýšila pořadové číslo zprávy, počínaje 1, bez mezer. IBM MQ zvyšuje pořadové číslo zprávy automaticky.
4. Vložila všechny logické zprávy do jedné skupiny zpráv, a teprve pak vložila logické zprávy do další skupiny zpráv.

Vzhledem k tomu, že aplikace sdělila správci front, jak vkládá zprávy do skupin a segmentů logických zpráv, aplikace nemusí udržovat a aktualizovat informace o skupinách a segmentech o každém volání MQPUT, protože správce front je uchovává a aktualizuje. Konkrétně to znamená, že aplikace nemusí nastavovat pole *GroupId*, *MsgSeqNumber*a *Offset* v MQMD, protože správce front nastavuje tato pole na příslušné hodnoty. Aplikace musí v produktu MQMD nastavit pouze pole *MsgFlags* , aby označovala, kdy zprávy patří do skupin nebo jsou segmenty logických zpráv, a označují poslední zprávu ve skupině nebo posledním segmentu logické zprávy.

Po spuštění skupiny zpráv nebo logické zprávy musí následná volání MQPUT určovat příslušné příznaky MQMF_ * v produktu *MsgFlags* v deskriptoru MQMD. Pokud se aplikace pokusí vložit zprávu, která není ve skupině, když existuje neukončená skupina zpráv, nebo pokud se jedná o neukončenou logickou zprávu, která není segmentem, volání selže s kódem příčiny MQRC_INCOMPLEE_GROUP nebo MQRC_INCOMPLEE_MSG, jak je to vhodné. Správce front však uchovává informace o aktuální skupině zpráv nebo aktuální logické zprávě a aplikace ji může ukončit odesláním zprávy (případně bez dat zprávy aplikace) zadáním volání MQMF_LAST_MSG_IN_GROUP nebo MQMF_LAST_SEGMENT před opětným zadáním volání MQPUT pro vložení zprávy, která není ve skupině, nebo se nejedná o segment.

Příkaz Obrázek 72 na stránce 744 zobrazuje kombinace voleb a příznaků, které jsou platné, a hodnoty polí *GroupId*, *MsgSeqNumber*a *Offset* , které správce front používá v každém případě. Kombinace voleb a příznaků, které nejsou zobrazeny v tabulce, jsou neplatné. Sloupce v tabulce mají následující významy: Either means Yes or No No:

PROTOKOL
Určuje, zda je v rámci volání zadána volba MQPMO_LOGICAL_ORDER.

MIGOCITY
Určuje, zda je v rámci volání zadána volba MQMF_MSG_IN_GROUP nebo MQMF_LAST_IN_GROUP.

SEGG
Určuje, zda je v rámci volání zadána volba MQMF_SEGMENT nebo MQMF_LAST_SEGMENT.

SEG OK
Určuje, zda je u volání zadána volba MQMF_SEGMENTATION_ALLOWED.

Cur grp
Určuje, zda před voláním existuje aktuální skupina zpráv.

Zpráva protokolu cur
Určuje, zda před voláním existuje aktuální logická zpráva.

Ostatní sloupce
Zobrazení hodnot, které správce front používá. Předchozí označuje hodnotu použitou pro pole v předchozí zprávě pro popisovač fronty.

Tabulka 102. Volby MQPUT související se zprávami ve skupinách a segmentech logických zpráv

Volby, které uvedete	Volby, které uvedete	Volby, které uvedete	Volby, které uvedete	Stav skupiny a protokolu-zpráva před voláním	Stav skupiny a protokolu-zpráva před voláním	Hodnoty, které správce front používá	Hodnoty, které správce front používá	Hodnoty, které správce front používá
PROTOKOL	MIGOCITY	SEGG	SEG OK	Cur grp	Zpráva protokolu cur	*GroupId*	*MsgSeqNumber*	*Offset*
Ano	Ne	Ne	Ne	Ne	Ne	MQGI_NONE	1	0

Tabulka 102. Volby MQPUT související se zprávami ve skupinách a segmentech logických zpráv (pokračování)

Volby, které uvedete	Volby, které uvedete	Volby, které uvedete	Volby, které uvedete	Stav skupiny a protokolu-zpráva před voláním	Stav skupiny a protokolu-zpráva před voláním	Hodnoty, které správce front používá	Hodnoty, které správce front používá	Hodnoty, které správce front používá
Ano	Ne	Ne	Ano	Ne	Ne	ID nové skupiny	1	0
Ano	Ne	Ano	buď	Ne	Ne	ID nové skupiny	1	0
Ano	Ne	Ano	buď	Ne	Ano	Předchozí ID skupiny	1	Předchozí odchylka + předchozí délka segmentu
Ano	Ano	buď	buď	Ne	Ne	ID nové skupiny	1	0
Ano	Ano	buď	buď	Ano	Ne	Předchozí ID skupiny	Předchozí pořadové číslo + 1	0
Ano	Ano	Ano	buď	Ano	Ano	Předchozí ID skupiny	Předchozí pořadové číslo	Předchozí odchylka + předchozí délka segmentu
Ne	Ne	Ne	Ne	buď	buď	MQGI_NONE	1	0
Ne	Ne	Ne	Ano	buď	buď	Nové ID skupiny, je-li hodnota MQGI_NONE, další hodnota v poli	1	0
Ne	Ne	Ano	buď	buď	buď	Nové ID skupiny, je-li hodnota MQGI_NONE, další hodnota v poli	1	Hodnota v poli
Ne	Ano	Ne	buď	buď	buď	Nové ID skupiny, je-li hodnota MQGI_NONE, další hodnota v poli	Hodnota v poli	0
Ne	Ano	Ano	buď	buď	buď	Nové ID skupiny, je-li hodnota MQGI_NONE, další hodnota v poli	Hodnota v poli	Hodnota v poli

Poznámka:

- Volání MQPMO_LOGICAL_ORDER není v rámci volání MQPUT1 platné.
- Pro pole *MsgId* správce front generuje nový identifikátor zprávy, je-li zadáno MQPMO_NEW_MSG_ID nebo MQMI_NONE, a v opačném případě použije hodnotu v poli.
- Pro pole *CorrelId* správce front vygeneruje nový korelační identifikátor, pokud je zadán parametr MQPMO_NEW_CORREL_ID a v opačném případě použije hodnotu v poli.

Určíte-li MQPMO_LOGICAL_ORDER, správce front vyžaduje, aby všechny zprávy ve skupině a segmenty v logické zprávě byly vloženy se stejnou hodnotou do pole *Persistence* v MQMD, tj. všechny musí být trvalé nebo všechny musí být přechodné. Není-li tato podmínka splněna, volání MQPUT selže s kódem příčiny MQRC_INCONSISTENT_PERSISTENCE.

Volba MQPMO_LOGICAL_ORDER má vliv na jednotky práce následujícím způsobem:

- Je-li první fyzická zpráva ve skupině nebo logické zprávě vložena do pracovní jednotky, musí být všechny ostatní fyzické zprávy ve skupině nebo v logické zprávě vloženy do jednotky práce, pokud se použije stejný popisovač fronty. Nepotřebují však být vloženy do stejné pracovní jednotky, což umožňuje skupině zpráv nebo logické zprávě, která se skládá z mnoha fyzických zpráv, které mají být rozděleny do dvou nebo více po sobě jdoucích jednotek práce pro manipulátor fronty.
- Pokud se první fyzická zpráva ve skupině nebo logické zprávě nevloží do pracovní jednotky, žádná z jiných fyzických zpráv ve skupině nebo logické zprávě nemůže být vložena do pracovní jednotky, pokud se použije stejný popisovač fronty.

Nejsou-li tyto podmínky splněny, volání MQPUT selže s kódem příčiny MQRC_INCONSISTENT_UOW.

Je-li zadán parametr MQPMO_LOGICAL_ORDER, MQMD zadaný v rámci volání MQPUT nesmí být menší než hodnota MQMD_VERSION_2. Není-li tato podmínka splněna, volání selže s kódem příčiny MQRC_WRONG_MD_VERSION.

Není-li uvedeno MQPMO_LOGICAL_ORDER, zprávy ve skupinách a segmentech logických zpráv lze vložit do libovolného pořadí a není nutné vkládat úplné skupiny zpráv ani úplné logické zprávy. Je odpovědností aplikace, aby zajistila, že pole *GroupId*, *MsgSeqNumber*, *Offset*a *MsgFlags* mají odpovídající hodnoty.

Tuto techniku použijte k restartování skupiny zpráv nebo logické zprávy uprostřed poté, co došlo k selhání systému. Když se systém restartuje, může aplikace nastavit pole *GroupId*, *MsgSeqNumber*, *Offset*, *MsgFlags*a *Persistence* na příslušné hodnoty a poté vydat volání MQPUT s parametrem MQPMO_SYNCPOINT nebo MQPMO_NO_SYNCPOINT, jak je požadováno, ale bez určení hodnoty MQPMO_LOGICAL_ORDER. Je-li toto volání úspěšné, uchovává správce front informace o skupině a segmentu a následná volání MQPUT používající tento manipulátor fronty mohou jako normální určit MQPMO_LOGICAL_ORDER.

Informace o skupinách a segmentech, které správce front uchovává pro volání MQPUT, jsou odděleny od informací o skupině a segmentu, které si zachovává pro volání MQGET.

Pro daný popisovač fronty může aplikace směšovat volání MQPUT, která určuje volání MQPMO_LOGICAL_ORDER s voláními MQPUT, ale povšimněte si následujících bodů:

- Není-li parametr MQPMO_LOGICAL_ORDER zadán, způsobí každé úspěšné volání MQPUT správce front tak, aby nastavil informace o skupině a segmentu pro manipulátor fronty na hodnoty zadané aplikací a nahradí existující informace o skupině a segmentech zachované správcem front pro manipulátor fronty.
- Není-li parametr MQPMO_LOGICAL_ORDER zadán, volání se nezdaří, pokud existuje aktuální skupina zpráv nebo logická zpráva; volání může být úspěšné s kódem dokončení MQCC_WARNING. Tabulka 103 na stránce 749 zobrazuje různé případy, které mohou nastat. V těchto případech, pokud kód dokončení není MQCC_OK, je kód příčiny jedním z následujících (je-li to vhodné):
  - SKUPINA MQRC_INCOMPLETE_GROUP
  - ZPRÁVA MQRC_INCOMPLETE_MSG
  - MQRC_INCONSISTENT_PERSISTENCE
  - NEKONZISTENCE MQRC_INCONSISTENT_UOW

  Poznámka: Správce front nekontroluje informace o skupině a segmentu pro volání MQPUT1 .

Tabulka 103. Výsledek, když volání MQPUT nebo MQCLOSE není konzistentní s informacemi o skupině a segmentu

Aktuální volání je	Předchozí volání bylo MQPUT s MQPMO_LOGICAL_ORDER	Předchozí volání bylo MQPUT bez MQPMO_LOGICAL_ORDER
MQPUT s MQPMO_LOGICAL_ORDER	SELHÁNÍ MQCC_FAILED	SELHÁNÍ MQCC_FAILED
MQPUT bez MQPMO_LOGICAL_ORDER	VAROVÁNÍ MQCC_WARNING	MQCC_OK
MQCLOSE s neukončené skupinou nebo logickou zprávou	VAROVÁNÍ MQCC_WARNING	MQCC_OK

Pro aplikace, které vložila zprávy a segmenty v logickém pořadí, zadejte MQPMO_LOGICAL_ORDER, protože je to nejjednodušší volba, která se má použít. Tato volba zbavuje aplikaci potřeby spravovat informace o skupinách a segmentech, protože tyto informace spravuje správce front. Nicméně specializované aplikace mohou vyžadovat větší kontrolu nad možností volby MQPMO_LOGICAL_ORDER, čehož lze dosáhnout neurčením této volby; pokud tak učiníte, musíte zajistit, aby pole *GroupId*, *MsgSeqNumber*, *Offset*a *MsgFlags* v MQMD byly nastaveny správně, a to před každým voláním MQPUT nebo MQPUT1 .

Například aplikace, která chce předat fyzickým zprávám, které přijímá, bez ohledu na to, zda se tyto zprávy nacházejí ve skupinách nebo segmentech logických zpráv, nesmí určovat MQPMO_LOGICAL_ORDER, a to ze dvou důvodů:

- Pokud jsou zprávy načteny a uvedeny v pořadí, určuje parametr MQPMO_LOGICAL_ORDER nový identifikátor skupiny pro zprávy, což může způsobit, že odesilateli zpráv může být obtížné nebo nemožné korelovat všechny zprávy odpovědi nebo zprávy, které jsou výsledkem skupiny zpráv.
- Ve složité síti s více cestami mezi odesílajícím a přijímajícím správcem front může dojít k nedostatku fyzických zpráv v pořadí. Neuvedení MQPMO_LOGICAL_ORDER a MQGMO_LOGICAL_ORDER na volání MQGET může pro každou fyzickou zprávu načíst a předat každou fyzickou zprávu, jakmile dorazí, aniž by čekal na příchod dalšího v logickém pořadí, aby se dospělo.

Aplikace, které generují zprávy sestav pro zprávy ve skupinách nebo segmentech logických zpráv, nesmí při vkládání zprávy sestavy uvádět také MQPMO_LOGICAL_ORDER.

MQPMO_LOGICAL_ORDER lze zadat s libovolněmi dalšími volbami MQPMO_ *.

Umístění logicky uspořádaných skupin do klastrované fronty (MQOO_BIND_ON_GROUP)

Volba MQOO_BIND_ON_OPEN zajišťuje, že všechny zprávy z této aplikace, a tedy všechny skupiny, jsou směrovány do jediné instance. To má nevýhodu, že provoz aplikací není vyrovnán přes více instancí fronty klastru. Chcete-li povolit vyrovnávání pracovní zátěže při zachování nedotčených skupin zpráv, je třeba nastavit následující volby:

- Volání MQPUT musí určovat MQPMO_LOGICAL_ORDER.
- Volání MQOPEN musí uvádět jednu z následujících dvou voleb:
  - SKUPINA MQO_BIND_ON_GROUP
  - MQOO_BIND_AS_Q_DEF a definice fronty musí určovat DEFBIND (GROUP)

Vyrovnávání pracovní zátěže je poté řízeno *mezi skupinami* zpráv bez nutnosti použití příkazů MQCLOSE a MQOPEN fronty. *Mezi skupinami* znamená, že proměnná MQMF_MSG_IN_GROUP je nastavena v deskriptoru MQMD (v2) nebo MQMDE a ve zpracování neexistuje žádná částečně dokončená skupina. Když probíhá skupina, budou vyřešený správce front a název fronty v manipulátoru objektu použity znovu.

Pokud byla předchozí zpráva nastavena na hodnotu MQPMO_LOGICAL_ORDER nebo MQMF_MSG_IN_GROUP, ale aktuální zpráva není součástí skupiny, volání PUT selže s hodnotou MQRC_INCOMPLEE_GROUP.

Pokud pro jednotlivé příkazy MQPUT není určen parametr MQPMO_LOGICAL_ORDER a žádná aktuální skupina není aktivní, bude pro tuto zprávu provedeno vyrovnávání pracovní zátěže (jako by volání MQOPEN určoval MQOO_BIND_NOT_FIXED).

Pro zprávy vázané na místo určení pomocí struktury MQOO_BIND_ON_GROUP není provedeno žádné opětovné přidělení. Další informace o opětovném přidělení naleznete v tématu “Skupiny zpráv” na stránce 38.

Seskupení logických zpráv

Pro použití logických zpráv ve skupině existují dva hlavní důvody:

- Je možné, že bude třeba zprávy zpracovat v určitém pořadí.
- Možná budete muset každou zprávu ve skupině zpracovat v souvisejícím způsobem.

V obou případech načtěte celou skupinu se stejnou instancí aplikace pro získání aplikace.

Předpokládejme například, že skupina se skládá ze čtyř logických zpráv. Aplikace bude vypadat takto:

```
PMO.Options = MQPMO_LOGICAL_ORDER | MQPMO_SYNCPOINT

MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
MQPUT MD.MsgFlags = MQMF_LAST_MSG_IN_GROUP

MQCMIT
```

Aplikace pro získání určuje volbu MQGMO_ALL_MSGS_AVAILABLE pro první zprávu ve skupině. Tím je zajištěno, že zpracování se nespustí, dokud nedorazíte všechny zprávy v rámci skupiny. Volba MQGMO_ALL_MSGS_AVAILABLE je ignorována pro následné zprávy ve skupině.

Když je načtena první logická zpráva skupiny, můžete použít příkaz MQGMO_LOGICAL_ORDER, abyste se ujistili, že zbývající logické zprávy skupiny jsou načteny v pořadí.

Takže aplikace pro získání vypadá takto:

```
/* Wait for the first message in a group, or a message not in a group */
GMO.Options = MQGMO_SYNCPOINT | MQGMO_WAIT
            | MQGMO_ALL_MSGS_AVAILABLE | MQGMO_LOGICAL_ORDER
do while ( GroupStatus == MQGS_MSG_IN_GROUP )
   MQGET
   /* Process each remaining message in the group */
   ...

MQCMIT
```

Další příklady seskupování zpráv viz “Segmentace aplikace logických zpráv” na stránce 761 a “Uvedení a získání skupiny, která zahrnuje jednotky práce” na stránce 750.

Informace o povolení požadavku na to, aby skupina zpráv byla alokována do stejné cílové instance pro fronty klastru, naleznete v části DefBind.

Uvedení a získání skupiny, která zahrnuje jednotky práce

V předchozím případě nemohou zprávy nebo segmenty začínat na opuštění uzlu (je-li jeho cíl vzdálený) nebo se má začít načítat, dokud není vložena celá skupina a jednotka práce je potvrzená. To nemusí být to, co chcete, pokud trvá dlouho, než se celá skupina, nebo je-li prostor fronty na uzlu omezený. K vyřešení tohoto stavu, dát skupinu do několika jednotek práce.

Je-li skupina vložena do více jednotek práce, je možné, aby se některé skupiny zavázaly, i když se aplikace nezdaří. Aplikace proto musí ukládat informace o stavu do každé jednotky práce, kterou může použít po restartování, aby mohla být obnovena nekompletní skupina. Nejjednodušší místo pro záznam těchto informací je ve frontě STATUS. Pokud byla úspěšně vložena úplná skupina, fronta STATUS je prázdná.

Je-li zahrnuta segmentace, logika je podobná. V tomto případě musí **StatusInfo** zahrnovat *Offset*.

Zde je příklad uvedení skupiny do několika jednotek práce:

```
PMO.Options = MQPMO_LOGICAL_ORDER | MQPMO_SYNCPOINT

/* First UOW */

MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
StatusInfo = GroupId,MsgSeqNumber from MQMD
MQPUT (StatusInfo to STATUS queue) PMO.Options = MQPMO_SYNCPOINT
MQCMIT

/* Next and subsequent UOWs */
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
MQGET (from STATUS queue) GMO.Options = MQGMO_SYNCPOINT
StatusInfo = GroupId,MsgSeqNumber from MQMD
MQPUT (StatusInfo to STATUS queue) PMO.Options = MQPMO_SYNCPOINT
MQCMIT

/* Last UOW */
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
MQPUT MD.MsgFlags = MQMF_LAST_MSG_IN_GROUP
MQGET (from STATUS queue) GMO.Options = MQGMO_SYNCPOINT
MQCMIT
```

Pokud byly potvrzeny všechny jednotky práce, celá skupina byla úspěšně vložena a fronta STATUS je prázdná. Pokud tomu tak není, skupina musí být znovu zahájena v bodě indikovaném informacemi o stavu. Objekt MQPMO_LOGICAL_ORDER nelze použít pro první vložení, ale poté jej lze použít.

Restart zpracování vypadá takto:

```
MQGET (StatusInfo from STATUS queue) GMO.Options = MQGMO_SYNCPOINT
if (Reason == MQRC_NO_MSG_AVAILABLE)
   /* Proceed to normal processing */
   ...

else
   /* Group was terminated prematurely */
   Set GroupId, MsgSeqNumber in MQMD to values from Status message
   PMO.Options = MQPMO_SYNCPOINT
   MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP

   /* Now normal processing is resumed.
      Assume this is not the last message */
   PMO.Options = MQPMO_LOGICAL_ORDER | MQPMO_SYNCPOINT
   MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
   MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP
   StatusInfo = GroupId,MsgSeqNumber from MQMD
   MQPUT (StatusInfo to STATUS queue) PMO.Options = MQPMO_SYNCPOINT
   MQCMIT
```

Od získání aplikace můžete začít zpracovávat zprávy ve skupině ještě před tím, než dorazila celá skupina. To zkracuje dobu odezvy ve zprávách v rámci skupiny, a také znamená, že úložiště není pro celou skupinu požadováno. Chcete-li si uvědomit výhody, použijte několik jednotek práce pro každou skupinu zpráv. Z důvodů zotavení je třeba načíst každou zprávu v rámci pracovní jednotky.

Stejně jako u příslušné aplikace vkládání je třeba zaznamenat informace o stavu, které mají být zaznamenávány automaticky, jakmile je každá jednotka práce potvrzená. Opět platí, že nejjednodušším místem pro záznam těchto informací je fronta STAVU. Pokud byla úplná skupina úspěšně zpracována, je fronta STATUS prázdná.

Poznámka: Pro intermediační jednotky práce se můžete vyhnout voláním MQGET z fronty STAVU tak, že každý z příkazů MQPUT do stavové fronty je segmentem zprávy (tj. nastavením příznaku MQMF_SEGMENT) místo toho, aby se pro každou jednotku práce zakládal úplná nová zpráva. V poslední pracovní jednotce je do stavové fronty vložen konečný segment s uvedením MQMF_LAST_SEGMENT a pak jsou informace o stavu vymazány s parametrem MQGET specifikací MQGMO_COMPLETE_MSG.

Během zpracování restartu místo použití jediné MQGET k získání možné stavové zprávy procházejte stavovou frontou s MQGMO_LOGICAL_ORDER, dokud nedosáhnete posledního segmentu (to znamená, dokud se nevrátí žádné další segmenty). V první pracovní jednotce po restartu také uveďte offset explicitně při umístění segmentu stavu.

V následujícím příkladu uvažujeme pouze o zprávách ve skupině za předpokladu, že vyrovnávací paměť aplikace je vždy dostatečně velká, aby udržela celou zprávu, ať už byla zpráva segmentována, či nikoli. MQGMO_COMPLETE_MSG je proto určen pro každý příkaz MQGET. Stejné zásady platí i pro segmentaci segmentace (v tomto případě musí StatusInfo obsahovat *Offset*).

Pro zjednodušení předpokládáme, že v rámci jediné jednotky UOW je načítána maximálně 4 zpráv:

```
msgs = 0    /* Counts messages retrieved within UOW */
/* Should be no status message at this point */

/* Retrieve remaining messages in the group */
do while ( GroupStatus == MQGS_MSG_IN_GROUP )

   /* Process up to 4 messages in the group */
   GMO.Options = MQGMO_SYNCPOINT | MQGMO_WAIT
               | MQGMO_LOGICAL_ORDER
   do while ( (GroupStatus == MQGS_MSG_IN_GROUP) && (msgs < 4) )
      MQGET
      msgs = msgs + 1
      /* Process this message */
      ...
   /* end while

   /* Have retrieved last message or 4 messages  */
   /* Update status message if not last in group */
   MQGET (from STATUS queue) GMO.Options = MQGMO_SYNCPOINT
   if ( GroupStatus == MQGS_MSG_IN_GROUP )
      StatusInfo = GroupId,MsgSeqNumber from MQMD
      MQPUT (StatusInfo to STATUS queue) PMO.Options = MQPMO_SYNCPOINT
   MQCMIT
   msgs = 0
/* end while

if ( msgs > 0 )
   /* Come here if there was only 1 message in the group */
   MQCMIT
```

Pokud byly potvrzeny všechny jednotky práce, celá skupina byla úspěšně načtena a fronta STATUS je prázdná. Pokud tomu tak není, skupina musí být znovu zahájena v bodě indikovaném informacemi o stavu. MQGMO_LOGICAL_ORDER nelze použít pro první načtení, ale poté jej lze použít.

Restart zpracování vypadá takto:

```
MQGET (from STATUS queue) GMO.Options = MQGMO_SYNCPOINT
if (Reason == MQRC_NO_MSG_AVAILABLE)
   /* Proceed to normal processing */
   ...

else
   /* Group was terminated prematurely */
   /* The next message on the group must be retrieved by matching
      the sequence number and group ID with those retrieved from the
      status information. */
   GMO.Options = MQGMO_COMPLETE_MSG | MQGMO_SYNCPOINT | MQGMO_WAIT
   MQGET GMO.MatchOptions = MQMO_MATCH_GROUP_ID | MQMO_MATCH_MSG_SEQ_NUMBER,
         MQMD.GroupId      = value from Status message,
         MQMD.MsgSeqNumber = value from Status message plus 1
   msgs = 1
   /* Process this message */
   ...

   /* Now normal processing is resumed */
   /* Retrieve remaining messages in the group */
   do while ( GroupStatus == MQGS_MSG_IN_GROUP )

      /* Process up to 4 messages in the group */
      GMO.Options = MQGMO_COMPLETE_MSG | MQGMO_SYNCPOINT | MQGMO_WAIT
                  | MQGMO_LOGICAL_ORDER
      do while ( (GroupStatus == MQGS_MSG_IN_GROUP) && (msgs < 4) )
```

```
            MQGET
            msgs = msgs + 1
            /* Process this message */
            ...

         /* Have retrieved last message or 4 messages  */
         /* Update status message if not last in group */
         MQGET (from STATUS queue) GMO.Options = MQGMO_SYNCPOINT
         if ( GroupStatus == MQGS_MSG_IN_GROUP )
            StatusInfo = GroupId,MsgSeqNumber from MQMD
            MQPUT (StatusInfo to STATUS queue) PMO.Options = MQPMO_SYNCPOINT
         MQCMIT
         msgs = 0
```

Získání konkrétní zprávy

Existuje celá řada způsobů, jak získat určitou zprávu z fronty. To jsou: výběr na MsgId a CorrelId, výběr na GroupId, MsgSeqČíslo a Posunutí a výběr na MsgToken. Řetězec výběru můžete použít také při otevření fronty.

Chcete-li získat určitou zprávu z fronty, použijte pole *MsgId* a *CorrelId* struktury MQMD. Aplikace však mohou tato pole explicitně nastavit, takže vámi zadané hodnoty nemusí identifikovat jedinečnou zprávu. Produkt Tabulka 104 na stránce 753 zobrazuje, jaká zpráva je načtena pro možná nastavení těchto polí. Tato pole jsou na vstupu ignorována, pokud určujete parametr MQGMO_MSG_UNDER_CURSOR v parametru **GetMsgOpts** volání MQGET.

Tabulka 104. Použití identifikátorů zpráv a korelací

Načtení ...	***MsgId***	***CorrelId***
První zpráva ve frontě	MQMI_NONE	MQCI_NONE
První zpráva, která odpovídá *MsgId*	Nenulový	MQCI_NONE
První zpráva, která odpovídá *CorrelId*	MQMI_NONE	Nenulový
První zpráva, která odpovídá produktu *MsgId* i *CorrelId*	Nenulový	Nenulový

V každém případě *první* znamená první zprávu splňující kritéria výběru (pokud není zadán parametr MQGMO_BRONEXT NEXT, znamená to, že se jedná o zprávu *další* v posloupnosti splňující kritéria výběru).

Při návratu nastaví volání MQGET pole *MsgId* a *CorrelId* na identifikátory zprávy a korelační identifikátory vrácené zprávy, pokud existují.

Nastavíte-li pole *Version* struktury MQMD na hodnotu 2, můžete použít pole *GroupId*, *MsgSeqNumbe*r a *Offset* . Produkt Tabulka 105 na stránce 753 zobrazuje, jaká zpráva je načtena pro možná nastavení těchto polí.

Tabulka 105. Použití identifikátoru skupiny

Načtení ...	**Volby shody**
První zpráva ve frontě	MQMO_NONE
První zpráva, která odpovídá *MsgId*	MQMO_MATCH_MSG_ID
První zpráva, která odpovídá *CorrelId*	MQMO_MATCH_CORREL_ID
První zpráva, která odpovídá *GroupId*	MQMO_MATCH_GROUP_ID
První zpráva, která odpovídá *MsgSeqNumber*	MQMO_MATCH_MSG_SEQ_NUMBER
První zpráva, která odpovídá *MsgToken*	MQMO_MATCH_MSG_TOKEN
První zpráva, která odpovídá *Offset*	MQMO_MATCH_OFSET

Notes:

1. Hodnota MQMO_MATCH_XXX znamená, že pole *XXX* ve struktuře MQMD je nastaveno na hodnotu, která má být porovnána.
2. Příznaky MQMO lze použít v kombinaci. Příklad: MQMO_MATCH_GROUP_ID, MQMO_MATCH_MSG_SEQ_NUMBER a MQMO_MATCH_OFFSET lze použít společně k poskytnutí segmentu identifikovaného pomocí polí *GroupId*, *MsgSeqNumber*a *Offset* .
3. Určíte-li MQGMO_LOGICAL_ORDER, bude ovlivněna zpráva, kterou se pokoušíte načíst, protože tato volba závisí na informacích o stavu řízených pro manipulátor fronty. Další informace o tomto tématu viz "Logické a fyzické uspořádání" na stránce 743 a Volby.

Volání MQGET obvykle načte první zprávu z fronty. Zadáte-li konkrétní zprávu při použití volání MQGET, musí správce front hledat frontu, dokud nenajde příslušnou zprávu. To může ovlivnit výkon vaší aplikace.

Pokud používáte strukturu MQGMO verze 2 nebo novější a neurčujte příznaky MQMO_MATCH_MSG_ID nebo MQMO_MATCH_CORREL_ID, není třeba pole *MsgId* nebo *CorrelId* mezi příkazy MQGET resetovat.

z/OS V systému IBM MQ for z/OSlze atribut fronty *IndexType* použít ke zvýšení rychlosti operací MQGET ve frontě. Další informace viz "Typ indexu" na stránce 758.

Konkrétní zprávu z fronty můžete získat zadáním jeho hodnoty MsgToken a MatchOption MQMO_MATCH_MSG_TOKEN v rámci struktury MQGMO. Hodnota MsgToken je vrácena voláním MQPUT, které původně danou zprávu umístily do fronty, nebo předchozími operacemi MQGET a zůstává konstantní, dokud nebude správce front restartován.

Zajímáte-li se pouze o podmnožinu zpráv ve frontě, můžete určit, které zprávy chcete zpracovat, pomocí výběrového řetězce s voláním MQOPEN nebo MQSUB. Funkce MQGET poté načte další zprávu, která odpovídá tomuto řetězci výběru. Další informace o výběrových řetězcích najdete v tématu "Selektory." na stránce 25.

Zlepšení výkonu přechodných zpráv

Když klient vyžaduje zprávu ze serveru, odešle požadavek na server. Odešle samostatný požadavek pro každou zprávu, kterou spotřebuje. Chcete-li zvýšit výkon klientů, kteří spotřebovávají přechodné zprávy, tím, že nechcete odesílat tyto zprávy požadavků, může být klient nakonfigurován tak, aby používal *čtení napřed*. Čtení napřed umožňuje odesílání zpráv klientovi, aniž by aplikace musela požadovat jejich zpracování.

Je-li povoleno čtení napřed, zprávy se posílají do vyrovnávací paměti na straně klienta s názvem *read ahead buffer*. Klient bude mít pro každou frontu dopředné vyrovnávací paměti čtení napřed s povoleným dopředným čtením. Zprávy ve vyrovnávací paměti dopředného čtení nejsou trvalé. Klient pravidelně aktualizuje server informacemi o množství dat, které spotřeboval.

Při volání MQOPEN s parametrem MQOO_READ_AHEAD povolí klient IBM MQ čtení napřed pouze v případě, že jsou splněny určité podmínky. Tyto podmínky zahrnují:

- Jak klient, tak i vzdálený správce front musí být IBM WebSphere MQ 7 nebo novější.
- Aplikace klienta musí být kompilována a propojena s použitím podprocesových knihoven klienta IBM MQ MQI.
- Kanál klienta musí používat protokol TCP/IP.
- Kanál musí mít nenulové nastavení SharingConversations (SHARECNV) v definici kanálu klienta i serveru.

Použití dopředného čtení může zlepšit výkon při spotřebovávání přechodných zpráv z klientské aplikace. Toto zlepšení výkonu je dostupné pro aplikace MQI i JMS . Klientské aplikace používající příkaz MQGET nebo asynchronní spotřeba budou těžit ze zlepšení výkonu při přijímání přechodných zpráv.

Ne všechny návrhy aplikací klienta jsou vhodné pro použití čtení napřed, protože nejsou podporovány všechny volby pro použití s dopředným čtením a některé volby musí být konzistentní mezi voláními MQGET, je-li povoleno dopředné čtení. Pokud klient změní svá kritéria výběru mezi voláními MQGET, zprávy uložené ve vyrovnávací paměti dopředného čtení zůstanou uvízlé v vyrovnávací paměti čtení napřed klienta.

Pokud již nejsou požadovány nevyřízené požadavky na uvízlé zprávy s předchozími výběrovými kritérii, lze na klientovi nastavit konfigurovatelný interval vymazání, aby se tyto zprávy od klienta automaticky vymazávaly. Interval vymazání je jedna ze skupiny voleb ladění dopředného čtení, kterou určuje klient. Tyto volby je možné vyladit tak, aby splňovaly vaše požadavky.

Je-li klientská aplikace restartována, mohou být zprávy v vyrovnávací paměti dopředného čtení ztraceny. A naopak, zpráva, která byla přesunuta do vyrovnávací paměti dopředného čtení, by mohla být odstraněna z podkladové fronty; nevzniká tím, že by byla z vyrovnávací paměti odebrána, takže volání MQGET s využitím dopředného čtení může vrátit zprávu, která již neexistuje.

Čtení napřed se provádí pouze pro vazby klienta. Atribut je ignorován pro všechny ostatní vazby.

Čtení napřed nemá žádný vliv na spuštění. Při dopředném čtení zprávy klientem není generována žádná zpráva spouštěče. Čtení napřed negeneruje informace evidence a statistiky, je-li povoleno.

Použití dopředného čtení s publikováním zasílání zpráv odběru

Když odběratelská aplikace určuje cílovou frontu, do které jsou publikace odesílána, použije se jako výchozí hodnota dopředného čtení hodnota DEFREADA zadané fronty.

Při požadavcích na odběry aplikací, které produkt IBM MQ spravuje místo určení, do kterého jsou publikování odesílána, je vytvořena spravovaná fronta jako dynamická fronta na základě předdefinované modelové fronty. Jedná se o hodnotu DEFREADA modelové fronty, která se používá jako výchozí hodnota dopředného čtení. Výchozí modelové fronty SYSTEM.DURABLE.PUBLICATIONS.MODEL nebo SYSTEM.NONDURABLE.PUBLICATIONS.MODEL se použije, pokud není definována modelová fronta pro toto nebo nadřízené téma.

Související pojmy

“Vyladění výkonu pro přechodné zprávy v systému AIX” na stránce 757
Pokud používáte produkt AIX 5.3 nebo novější, zvažte nastavení parametru ladění pro použití plného výkonu pro přechodné zprávy.

Související úlohy

“Zapnutí a vypnutí dopředného čtení” na stránce 756
Při výchozím nastavení je dopředné čtení vypnuto. Čtení napřed můžete povolit na úrovni fronty nebo aplikace.

Související odkazy

“Volby MQGET a dopředné čtení” na stránce 755
Je-li povoleno čtení napřed, nejsou podporovány všechny volby MQGET; některé volby jsou vyžadovány k konzistenci mezi voláními MQGET.

Volby MQGET a dopředné čtení
Je-li povoleno čtení napřed, nejsou podporovány všechny volby MQGET; některé volby jsou vyžadovány k konzistenci mezi voláními MQGET.

Při volání MQOPEN s parametrem MQOO_READ_AHEAD povolí klient IBM MQ čtení napřed pouze v případě, že jsou splněny určité podmínky. Tyto podmínky zahrnují:

- Jak klient, tak i vzdálený správce front musí být IBM WebSphere MQ 7 nebo novější.
- Aplikace klienta musí být kompilována a propojena s použitím podprocesových knihoven klienta IBM MQ MQI.
- Kanál klienta musí používat protokol TCP/IP.
- Kanál musí mít nenulové nastavení SharingConversations (SHARECNV) v definici kanálu klienta i serveru.

Následující tabulka uvádí, které volby jsou podporovány pro použití s dopředným čtením a zda mohou být změněny mezi voláními MQGET.

Tabulka 106. Volby MQGET a dopředné čtení			
	Povoleno, je-li dopředné čtení povoleno a lze je měnit mezi voláními MQGET [5]	**Povoleno, je-li dopředné čtení povoleno, ale nelze je měnit mezi voláními MQGET [1]**	**Volby MQGET, které nejsou povoleny, je-li povoleno čtení napřed [2]**
MQGET MQMD	MsgId [3] CorrelId [3]	Kódování CodedCharSetId	
Volby MQGMO MQGET	• MQGMO_NO_WAIT • MQGMO_BROWSE_MESSAGE _UNDER_CURSOR • NEJPRVE MQGMO_BROWSE_FIRST • PŘÍŠTĚ MQGMO_BROWSE_NEXT • FUNKCE MQGMO_FAIL_IF_QUIESCING	• MQGMO_SYNCPOINT_IF _PERSISTENT • MQGMO_NO_SYNCPOINT • MQGMO_ACCEPT_ ZKRÁCENÁ _ZPRÁVA • MQGMO_CONVERT	• SIGNÁL MQGMO_SET_DATA • MQGMO_SYNCPOINT • MQGMO_MARKER_SKIP _BACKOUT • MQGMO_MSG_UNDER _CURSOR [4] • MQGMOVÝ_ZÁMEK • MQGMO_ODEMKNOUT • MQGMO_LOGICAL_ORDER • ZPRÁVA MQGMO_COMPLETE_MSG • MQGMO_ALL_MSGS_AVAILABLE • MQGMO_ALL_SEGMENTS_ K DISPOZICI

Notes:

1. Pokud jsou tyto volby upraveny mezi voláními MQGET, je vrácen kód příčiny MQRC_OPTIONS_CHANGED.
2. Pokud se tyto volby zadaly při prvním volání MQGET, bude dopředné čtení zablokováno. Budou-li tyto volby zadány při následném volání MQGET, vrátí se kód příčiny MQRC_OPTIONS_ERROR.
3. Pokud aplikace klienta změní hodnoty MsgId a CorrelId mezi voláními MQGET, zprávy s předchozími hodnotami již mohly být odeslány klientovi a zůstanou ve vyrovnávací paměti pro čtení napřed klienta, dokud nebudou spotřebovány (nebo automaticky vymazány).
4. MQGMO_MSG_UNDER_CURSOR nelze použít s dopředným čtením. Čtení napřed je zakázáno, když jsou při otevření fronty určeny volby MQOO_BROWSE a jeden z voleb MQOO_INPUT_SHARED nebo MQOO_INPUT_EXCLUSIVE.
5. Je-li dopředné čtení povoleno, první příkaz MQGET určuje, zda mají být zprávy procházeny nebo načíteny z fronty. Pokud klientská aplikace poté použije příkaz MQGET se změněnými volbami, jako například při pokusu o procházení po počátečním získání nebo při pokusu o získání následujícího počátečního procházení, je vrácen kód příčiny MQRC_OPTIONS_CHANGED.

Pokud klient změní svá kritéria výběru mezi voláními MQGET, zprávy uložené v vyrovnávací paměti dopředného čtení, které odpovídají počátečnímu kritériu výběru, nejsou aplikacemi klienta spotřebovávány a zůstávají uvízlé v paměti dopředného čtení klienta. V situacích, kdy vyrovnávací paměť pro čtení napřed klienta obsahuje mnoho uvízlých zpráv, výhody spojené s dopředným čtením se ztratí a pro každou spotřebovanou zprávu se vyžaduje zvláštní požadavek na server. Chcete-li určit, zda je čtení napřed používáno efektivně, můžete použít parametr stavu připojení, READA.

Čtení napřed může být zablokované, je-li to požadováno aplikací kvůli nekompatibilním volbám uvedeným na prvním volání MQGET. V této situaci stav připojení ukazuje dopředné čtení jako zablokované.

Pokud se kvůli těmto omezením na MQGET rozhodnete, že návrh aplikace klienta není vhodný pro čtení napřed, určete volbu MQOPEN MQOO_READ_AHEAD_NO. Nebo nastavte výchozí hodnotu dopředného čtení u fronty, která se otevírá, změněna na hodnotu NO nebo DISABLED.

Zapnutí a vypnutí dopředného čtení
Při výchozím nastavení je dopředné čtení vypnuto. Čtení napřed můžete povolit na úrovni fronty nebo aplikace.

Informace o této úloze

Při volání MQOPEN s parametrem MQOO_READ_AHEAD povolí klient IBM MQ čtení napřed pouze v případě, že jsou splněny určité podmínky. Tyto podmínky zahrnují:

- Jak klient, tak i vzdálený správce front musí být IBM WebSphere MQ 7 nebo novější.
- Aplikace klienta musí být kompilována a propojena s použitím podprocesových knihoven klienta IBM MQ MQI.

- Kanál klienta musí používat protokol TCP/IP.
- Kanál musí mít nenulové nastavení SharingConversations (SHARECNV) v definici kanálu klienta i serveru.

Povolení dopředného čtení:

- Chcete-li konfigurovat dopředné čtení na úrovni fronty, nastavte atribut queue, DEFREADA na YES.
- Chcete-li konfigurovat čtení napřed na úrovni aplikace, postupujte takto:
  - chcete-li při volání funkce MQOPEN používat funkci MQOO_READ_AHEAD, je-li to možné, použijte volbu dopředného čtení. Pokud byl atribut fronty DEFREADA nastaven na hodnotu DISABLED, není možné, aby aplikace klienta používala čtení napřed.
  - chcete-li používat dopředné čtení pouze v případě, že je ve frontě povoleno čtení napřed, použijte volbu MQOO_READ_AHEAD_AS_Q_DEF u volání funkce MQOPEN.

Není-li návrh klientské aplikace vhodný pro čtení napřed, můžete jej zakázat:

- na úrovni fronty nastavením atributu fronty DEFREADA na NO, pokud nechcete použít dopředné čtení, pokud není vyžádáno klientskou aplikací, nebo VYPNUTÉ, pokud nechcete, aby bylo čtení napřed používáno bez ohledu na to, zda je dopředné čtení vyžadováno klientskou aplikací.
- na úrovni aplikace pomocí volby MQOO_NO_READ_AHEAD při volání funkce MQOPEN.

Dvě volby MQCLOSE vám umožňují konfigurovat, co se stane se všemi zprávami, které jsou ukládány do vyrovnávací paměti dopředného čtení, pokud je fronta uzavřena.

- Použijte MQCO_IMMEDIATE k zahození zpráv do vyrovnávací paměti dopředného čtení.
- Použijte MQCO_QUIESCE, abyste zajistili, že zprávy ve vyrovnávací paměti dopředného čtení budou spotřebovány aplikací před zavřením fronty. Je-li zadán příkaz MQCLOSE s hodnotou MQCO_QUIESCE a existují zprávy, které zbývají do vyrovnávací paměti čtení napřed, funkce MQRC_READ_AHEAD_MSGS se vrátí s hodnotou MQCC_WARNING.

Vyladění výkonu pro přechodné zprávy v systému AIX
Pokud používáte produkt AIX 5.3 nebo novější, zvažte nastavení parametru ladění pro použití plného výkonu pro přechodné zprávy.

Chcete-li nastavit parametr ladění tak, aby byl účinný okamžitě, zadejte jako uživatel root následující příkaz:

```
/usr/sbin/ioo -o j2_nPagesPerWriteBehindCluster=0
```

Chcete-li nastavit parametr ladění tak, aby se okamžitě nabyl účinku a přetrvá po opětovném zavedení systému, zadejte jako uživatel root následující příkaz:

```
/usr/sbin/ioo -p -o j2_nPagesPerWriteBehindCluster=0
```

Normálně se přechodné zprávy uchovávají pouze v paměti, ale existují okolnosti, kdy produkt AIX může naplánovat zápis přechodných zpráv na disk. Zprávy, které mají být zapsány na disk, jsou nedostupné pro příkaz MQGET až do dokončení zápisu na disk. Navrhovaný příkaz vyladění tuto prahovou hodnotu liší; místo plánování zápisu zpráv na disk, je-li ve frontě uvedeno 16 kilobajtů dat, dochází k zápisu na disk pouze v případě, že se skutečné úložiště na počítači blíží plné. To je globální změna a může ovlivnit další softwarové komponenty.

On AIX, when using multithreaded applications and especially when running on machines with multiple processors, we strongly recommend setting AIXTHREAD_SCOPE=S in the mqm ID `.profile` or setting AIXTHREAD_SCOPE=S in the environment before starting the application, for better performance and more solid scheduling. Příklad:

```
export AIXTHREAD_SCOPE=S
```

Nastavení AIXTHREAD_SCOPE=S znamená, že uživatelské podprocesy vytvořené s výchozími atributy jsou umístěny do rozsahu soupeření celého systému. Je-li uživatelský podproces vytvořen s rozsahem soupeření celého systému, je svázán s vláknem jádra a je naplánován jádrem. Základní podproces jádra není sdílen s žádným jiným uživatelským podprocesem.

Deskriptory souborů

Při spuštění vícevláknového procesu, jako je proces agenta, můžete dosáhnout měkkého limitu pro deskriptory souborů. Tento limit uvádí kód příčiny IBM MQ `MQRC_UNEXPECTED_ERROR (2195)` a v případě, že je zde dostatek deskriptorů souborů, soubor IBM MQ FFST™ .

Chcete-li se tomuto problému vyhnout, můžete zvýšit limit procesu pro počet deskriptorů souboru. Chcete-li to provést, změňte atribut `nofiles` v `/etc/security/limits` na 10.000 pro ID uživatele mqm nebo ve výchozí stanze.

Limity systémových prostředků

Nastavte omezení systémových prostředků pro segment dat a segment zásobníku na neomezený počet pomocí následujících příkazů v příkazovém řádku:

```
ulimit -d unlimited
ulimit -s unlimited
```

Typ indexu

Atribut fronty, *IndexType*, určuje typ indexu, který správce front udržuje za účelem zvýšení rychlosti operací MQGET ve frontě.

Poznámka: Podporováno pouze na IBM MQ for z/OS.

Máte pět možností:

Hodnota	Popis
ŽÁDNÉ	Není udržován žádný index. Tuto volbu použijte při sekvenčním načítání zpráv (viz “Priorita” na stránce 742).
groupID	Je udržován index identifikátorů skupin. Tento typ indexu je třeba použít, pokud chcete logické pořadí skupin zpráv (viz “Logické a fyzické uspořádání” na stránce 743).
MSGID	Je udržován index identifikátorů zpráv. Tuto možnost použijte při načítání zpráv s použitím pole *MsgId* jako kritéria výběru na volání MQGET (viz “Získání konkrétní zprávy” na stránce 753).
MsgToken	Je udržován index tokenů zpráv.
CorrelId	Je udržován index identifikátorů korelace. Tuto možnost použijte při načítání zpráv s použitím pole *CorrelId* jako kritéria výběru na volání MQGET (viz “Získání konkrétní zprávy” na stránce 753).

Poznámka:

1. Pokud provádíte indexaci pomocí volby MSGID nebo volby CORRELID, nastavte relativní parametry **MsgId** nebo **CorrelId** v deskriptoru MQMD. Není výhodné nastavit obojí.
2. Procházení používá mechanizmus indexu k nalezení zprávy v případě, že fronta vyhovuje všem následujícím podmínkám:
   - Má typ index MSGID, CORRELID, nebo GROUPID
   - Je procházen se stejným typem ID
   - Má zprávy pouze jedné priority
3. Vyvarovat se front (indexovaných podle *MsgId* nebo *CorrelId*) obsahující tisíce zpráv, protože to ovlivní čas restartování. (To neplatí pro přechodné zprávy, protože jsou odstraněny při restartu.)

4. MSGTOKEN se používá k definování front spravovaných správcem pracovní zátěže z/OS .

Úplný popis atributu **IndexType** naleznete v části IndexType. Další informace o atributu **IndexType** viz “Aspekty návrhu a výkonu pro aplikace produktu z/OS” na stránce 53.

Zpracování zpráv větších než 4 MB

Zprávy mohou být příliš velké pro aplikaci, frontu nebo správce front. V závislosti na prostředí nabízí produkt IBM MQ řadu způsobů, jak se vypořádat se zprávami, které jsou delší než 4 MB.

Můžete zvýšit atribut **MaxMsgLength** až o 100 MB na všech systémech IBM MQ na úrovni V6 nebo vyšší. Nastavte tuto hodnotu tak, aby odrážela velikost zpráv používajících frontu. V systémech IBM MQ jiných než IBM MQ for z/OSmůžete také:

1. Použijte segmentované zprávy. (Zprávy mohou být segmentovány buď aplikací, nebo správcem front.)
2. Použít referenční zprávy.

Každý z těchto přístupů je popsán ve zbývající části této části.

Zvýšení maximální délky zprávy

Atribut správce front **MaxMsgLength** definuje maximální délku zprávy, kterou může správce front zpracovat. Podobně atribut fronty **MaxMsgLength** je maximální délka zprávy, kterou lze zpracovat pomocí fronty. Podporovaná výchozí maximální délka zprávy závisí na prostředí, ve kterém pracujete.

Pokud obsluhujete velké zprávy, můžete tyto atributy změnit nezávisle na jiných platformách než z/OS. Hodnotu atributu správce front lze nastavit v rozsahu 32768 bajtů až 100 MB.

Upozornění: V systému IBM MQ for z/OS je atribut **MaxMsgLength** správce front pevně naprogramován na 100 MB.

Na všech platformách můžete nastavit hodnotu atributu fronty v rozsahu 0 až 100 MB.

Poté, co změníte jeden nebo oba atributy **MaxMsgLength** , restartujte aplikace a kanály, abyste se ujistili, že se změny projeví.

Po provedení těchto změn musí být délka zprávy menší nebo rovna než atributy fronty a správce front **MaxMsgLength** . Existující zprávy však mohou být delší než jeden z těchto atributů.

Je-li zpráva pro frontu příliš velká, vrátí se hodnota MQRC_MSG_TOO_BIG_FOR_Q. Podobně, je-li zpráva pro správce front příliš velká, vrátí se hodnota MQRC_MSG_TOO_BIG_FOR_Q_MGR.

Tato metoda manipulace s velkými zprávami je snadná a pohodlná. Před použitím však zvažte následující faktory:

- Jednotnost správců front se snižuje. Maximální velikost dat zprávy je určena serverem *MaxMsgLength* pro každou frontu (včetně přenosových front), na které bude zpráva vložena. Tato hodnota je často standardně nastavena na hodnotu *MaxMsgLength*správce front, zvláště pro přenosové fronty. Proto je obtížné předpovídat, zda je zpráva příliš velká, když má cestovat do vzdáleného správce front.
- Využití systémových prostředků se zvýšilo. Aplikace například potřebují větší vyrovnávací paměti a na některých platformách může dojít k vyššímu využití sdíleného úložiště. Úložný prostor fronty by měl být ovlivněn pouze tehdy, je-li to požadováno pro větší zprávy.
- Postižení kanálu kanálu je ovlivněno. Velká zpráva je stále započítává pouze jako jedna zpráva k počtu dávek, ale potřebuje delší dobu přenosu, čímž se zvyšuje doba odezvy pro ostatní zprávy.

Multi *Segmentace zpráv*

Tyto informace použijte k seznámení se s segmentováním zpráv. Tato funkce není v produktu IBM MQ for z/OS nebo v aplikacích využívajících produkt IBM MQ classes for JMSpodporována.

Zvýšení maximální délky zprávy, jak je vysvětleno v tématu “Zvýšení maximální délky zprávy” na stránce 759 , má určité negativní důsledky. Tato zpráva může také způsobit, že zpráva je příliš velká pro frontu nebo správce front. V těchto případech můžete segmentovat zprávu. Další informace o segmentech viz “Skupiny zpráv” na stránce 38.

Další sekce se podívejte na běžné použití pro segmentování zpráv. Při získávání a destruktivním získávání se předpokládá, že volání MQPUT nebo MQGET *vždy* fungují v rámci pracovní jednotky. Vždy zvažte použití této techniky, aby se snížila možnost přítomnosti neúplných skupin v síti. Předpokládá se jednofázové potvrzení správce front, ale jiné metody koordinace jsou stejně platné.

Kromě toho se při získávání aplikací předpokládá, že pokud více serverů zpracovává stejnou frontu, každý server zpracuje podobný kód, takže jeden server nikdy neselže při hledání zprávy nebo segmentu, které očekává, že tam bude (protože bylo zadáno MQGMO_ALL_MSGS_AVAILABLE nebo MQGMO_ALL_SEGMENTS_AVAILABLE).

Vložení a získání segmentované zprávy, která obsahuje jednotky práce

Můžete vložit a získat segmentovanou zprávu, která se rozprostíří do jednotky práce podobným způsobem jako “Uvedení a získání skupiny, která zahrnuje jednotky práce” na stránce 750.

Nemůžete však vložit nebo načíst segmentované zprávy v rámci globální transakce.

Multi *Segmentace a opětovné sestavení pro správce front*

Jedná se o nejjednodušší scénář, ve kterém jedna aplikace vkládá zprávu, která má být načtena jinou aplikací. Zpráva může být velká: není příliš velká pro vložení nebo získání aplikace na zpracování v jedné vyrovnávací paměti, ale příliš velká pro správce front nebo frontu, na kterou má být zpráva vložena.

Jediné změny, které jsou nezbytné pro tyto aplikace, jsou určeny k tomu, aby aplikace mohla autorizovat správce front, aby provedl segmentaci, je-li to nutné:

```
   PMO.Options = (existing options)
   MD.MsgFlags = MQMF_SEGMENTATION_ALLOWED
   MD.Version = MQMD_VERSION_2
   memcpy(MD.GroupId, MQGI_NONE, MQ_GROUP_ID_LENGTH)
MQPUT
```

a v případě aplikace, která má požádat správce front o opětné sestavení zprávy, pokud byla segmentována:

```
   GMO.Options = MQGMO_COMPLETE_MSG | (existing options)
   MQGET
```

V tomto nejjednodušším scénáři musí aplikace před voláním MQPUT resetovat pole GroupId na hodnotu MQGI_NONE, aby správce front mohl generovat jedinečný identifikátor skupiny pro každou zprávu. Pokud to není provedeno, nespřízněné zprávy mohou mít stejný identifikátor skupiny, což může následně vést k nesprávnému zpracování.

Vyrovnávací paměť aplikace musí být dostatečně velká, aby mohla obsahovat znovu sestavovanou zprávu (pokud nezahrnete volbu MQGMO_ACCEPT_TRUNCATED_MSG).

Pokud má být atribut MAXMSGLEN fronty upraven tak, aby pojmul segmentaci zpráv, pak zvažte:

- Minimální segment zpráv podporovaný v lokální frontě je 16 bajtů.
- Pro přenosovou frontu musí parametr MAXMSGLEN také obsahovat prostor potřebný pro záhlaví. Zvažte použití hodnoty o velikosti alespoň 4000 bajtů větší než maximální očekávaná délka uživatelských dat v libovolném segmentu zprávy, který lze vložit do přenosové fronty.

Je-li konverze dat nezbytná, může ji aplikace vyžadovat zadáním hodnoty MQGMO_CONVERT. To by mělo být jednoduché, protože uživatelská procedura pro převod dat se zobrazí spolu s úplnou zprávou. Nepokoušejte se převést data v odesílacím kanálu, je-li zpráva segmentovaná, a formát dat je takový, že uživatelská procedura konverze dat nemůže provést převod na neúplných datech.

Multi *Segmentace aplikace*

Segmentace aplikace se používá tehdy, když segmentace správce front není vhodná, nebo když aplikace vyžadují konverzi dat se specifickými hranicemi segmentu.

Segmentace aplikace se používá ze dvou hlavních důvodů:

1. Segmentace správce front sama o sobě není adekvátní, protože zpráva je příliš velká, aby ji bylo možné zpracovat v jedné vyrovnávací paměti aplikacemi.
2. Převod dat musí být proveden odesílacími kanály a formát je takový, že aplikace musí stanovit, kde mají být hranice segmentu, aby mohly být konverze jednotlivých segmentů možné.

Pokud však převod dat není problém, nebo pokud aplikace získání vždy používá MQGMO_COMPLETE_MSG, segmentaci správce front lze také povolit zadáním hodnoty MQMF_SEGMENTATION_ALLOWED. V našem příkladu aplikace segmentuje zprávu do čtyř segmentů:

```
PMO.Options = MQPMO_LOGICAL_ORDER | MQPMO_SYNCPOINT

MQPUT MD.MsgFlags = MQMF_SEGMENT
MQPUT MD.MsgFlags = MQMF_SEGMENT
MQPUT MD.MsgFlags = MQMF_SEGMENT
MQPUT MD.MsgFlags = MQMF_LAST_SEGMENT

MQCMIT
```

Pokud nepoužíváte MQPMO_LOGICAL_ORDER, aplikace musí nastavit *Offset* a délku každého segmentu. V tomto případě se logický stav neudržuje automaticky.

Aplikace pro získání nemůže zaručit, že bude mít vyrovnávací paměť dostatečně velkou vyrovnávací paměť, aby mohla být zadržena znovu sestavená zpráva. Musí být proto připraven zpracovávat jednotlivé segmenty jednotlivě.

U zpráv, které jsou segmentovány, nechce tato aplikace začít zpracovávat jeden segment, dokud nejsou přítomny všechny segmenty tvořící logickou zprávu. Funkce MQGMO_ALL_SEGMENTS_AVAILABLE je proto určena pro první segment. Zadáte-li MQGMO_LOGICAL_ORDER a aktuální logická zpráva, bude hodnota MQGMO_ALL_SEGMENTS_AVAILABLE ignorována.

Po načtení prvního segmentu logické zprávy použijte MQGMO_LOGICAL_ORDER, abyste se ujistili, že zbývající segmenty logické zprávy jsou načteny v pořadí.

Ve zprávách v různých skupinách není věnována žádná pozornost. Pokud se takové zprávy vyskytnou, jsou zpracovány v pořadí, ve kterém se první segment každé zprávy vyskytuje ve frontě.

```
GMO.Options = MQGMO_SYNCPOINT | MQGMO_LOGICAL_ORDER
            | MQGMO_ALL_SEGMENTS_AVAILABLE | MQGMO_WAIT
do while ( SegmentStatus == MQSS_SEGMENT )
   MQGET
   /* Process each remaining segment of the logical message */
   ...

MQCMIT
```

Multi *Segmentace aplikace logických zpráv*

Zprávy musí být udržovány v logickém pořadí ve skupině a některé nebo všechny z nich by mohly být tak velké, že vyžadují segmentaci aplikace.

V našem příkladu má být vložena skupina čtyř logických zpráv. Všechny kromě třetí zprávy jsou velké a vyžadují segmentaci, kterou provádí aplikace vkládání:

```
PMO.Options = MQPMO_LOGICAL_ORDER | MQPMO_SYNCPOINT

MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP      | MQMF_SEGMENT
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP      | MQMF_SEGMENT
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP      | MQMF_LAST_SEGMENT

MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP      | MQMF_SEGMENT
MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP      | MQMF_LAST_SEGMENT

MQPUT MD.MsgFlags = MQMF_MSG_IN_GROUP

MQPUT MD.MsgFlags = MQMF_LAST_MSG_IN_GROUP | MQMF_SEGMENT
MQPUT MD.MsgFlags = MQMF_LAST_MSG_IN_GROUP | MQMF_SEGMENT
MQPUT MD.MsgFlags = MQMF_LAST_MSG_IN_GROUP | MQMF_LAST_SEGMENT
```

```
MQCMIT
```

V aplikaci GET je MQGMO_ALL_MSGS_AVAILABLE uveden v první MQGET. To znamená, že se nenačtou žádné zprávy nebo segmenty skupiny, dokud nebude k dispozici celá skupina. Když byla načtena první fyzická zpráva skupiny, je použit MQGMO_LOGICAL_ORDER, aby se zajistilo, že segmenty a zprávy skupiny se načtou v pořadí:

```
GMO.Options = MQGMO_SYNCPOINT | MQGMO_LOGICAL_ORDER
            | MQGMO_ALL_MSGS_AVAILABLE | MQGMO_WAIT

do while ( (GroupStatus   != MQGS_LAST_MSG_IN_GROUP) ||
             (SegmentStatus != MQGS_LAST_SEGMENT) )
   MQGET
   /* Process a segment or complete logical message.  Use the GroupStatus
      and SegmentStatus information to see what has been returned */
   ...

MQCMIT
```

Poznámka: Pokud uvedete MQGMO_LOGICAL_ORDER a existuje aktuální skupina, MQGMO_ALL_MSGS_AVAILABLE se ignoruje.

Referenční zprávy
Tyto informace použijte k získání více informací o referenčních zprávách.

Poznámka: Není podporováno v produktu IBM MQ for z/OS.

Tato metoda umožňuje přenos velkého objektu z jednoho uzlu do jiného bez uložení objektu ve frontách produktu IBM MQ ve zdrojovém nebo v cílovém uzlu. To je zvláště výhodné v případě, že data existují v jiné formě, například pro poštovní aplikace.

Chcete-li tak učinit, zadejte uživatelskou proceduru pro zprávy na obou koncích kanálu. Další informace o tom, jak to provést, viz “Ukončovací programy zpráv kanálu” na stránce 944.

IBM MQ definuje formát záhlaví referenční zprávy (MQRMH). Popis naleznete v tématu MQRMH . Tato hodnota je rozpoznána s definovaným názvem formátu a může být následována skutečnými daty.

Chcete-li zahájit přenos velkého objektu, aplikace může vložit zprávu obsahující záhlaví referenční zprávy bez dat, která následují. Když tato zpráva opustí uzel, výstupní program zprávy načte příslušný objekt vhodným způsobem a připojí jej k referenční zprávě. Pak vrátí zprávu (nyní větší než dříve) odesílajícímu agentovi MCA pro přenos do přijímajícího agenta MCA.

Další uživatelská procedura pro zprávy je konfigurována v přijímajícím agentu MCA. Když tento výstup zprávy přijme jednu z těchto zpráv, vytvoří objekt s pomocí dat objektu, která byla přidána, a předá na referenční zprávu *bez* . Referenční zpráva může být nyní přijata aplikací a tato aplikace ví, že objekt (nebo alespoň část zastupovaného touto referenční zprávou) byl vytvořen v tomto uzlu.

Maximální množství dat objektů, které může odesílající uživatelská procedura zprávy připojit k referenční zprávě, je omezena vyjednanou maximální délkou zprávy pro kanál. Ukončení může vrátit pouze jednu zprávu pro práci s agentem MCA pro každou zprávu, že je předána, takže aplikace umisťování může vložit několik zpráv, aby mohl být jeden objekt přenesen. Každá zpráva musí identifikovat *logickou* délku a posun objektu, který má být k němu připojen. Avšak v případech, kdy není možné zjistit celkovou velikost objektu nebo maximální velikost povolenou kanálem, navrhněte odesílající uživatelskou proceduru tak, aby vkládaná aplikace vkládala pouze jednu zprávu a uživatelská procedura umístí další zprávu do přenosové fronty, jakmile ji přidá tolik dat, kolik může být posláno ke zprávě.

Před použitím této metody práce s velkými zprávami zvažte následující body:

- Agent MCA a uživatelská procedura pro zprávy jsou spuštěny pod ID uživatele produktu IBM MQ . Uživatelská procedura zprávy (a proto ID uživatele) potřebuje mít přístup k objektu, aby jej buď získal na odesílající straně, nebo jej vytvořil na přijímajícím konci; to může být proveditelné pouze v případech, kdy je objekt univerzálně přístupný. Vzniká tak otázka zabezpečení.

- Pokud se referenční zpráva s hromadně připojenými daty musí projít několika správci front, než dosáhne svého cíle, jsou hromadné údaje přítomny ve frontách produktu IBM MQ na uzlech, které zasahují do těchto uzlů. V těchto případech však není třeba žádná zvláštní podpora nebo východy.
- Návrh ukončení zprávy bude ztížit, je-li povoleno přesměrování nebo řazení do fronty s dead-letter. V těchto případech může dojít k tomu, že se části objektu dostanou mimo pořadí.
- Když na místo určení dorazí referenční zpráva, přijímající uživatelská procedura zprávy vytvoří objekt. Avšak, toto není synchronizováno s pracovní jednotkou MCA, takže pokud je dávka vrácena, další referenční zpráva obsahující tuto stejnou část objektu bude doručena do pozdější dávky a uživatelská procedura zprávy se může pokusit znovu vytvořit stejnou část objektu. Je-li objekt například řada aktualizací databáze, může být tato situace nepřijatelná. Je-li tomu tak, uživatelská procedura pro zprávy musí uchovávat protokol o tom, které aktualizace byly použity; to může vyžadovat použití fronty IBM MQ .
- V závislosti na charakteristice typu objektu může dojít k tomu, že zpráva se ukončí a aplikace budou muset spolupracovat při zachování počtu použití, aby mohl být objekt vymazán, když již není potřeba. Identifikátor instance může být také povinný; v záhlaví zprávy odkazu je k dispozici pole (viz MQRMH).
- Je-li jako distribuční seznam vložena referenční zpráva, musí být objekt vyhledatelný pro každý výsledný distribuční seznam nebo cílové místo určení v daném uzlu. Je možné, že budete potřebovat zachovat počty použití. Zvažte také možnost, že by uzel mohl být konečným uzlem pro některá místa určení v seznamu, ale zprostředkující uzel pro ostatní.
- Hromadná data se obvykle nekonvertují. Důvodem je to, že ke konverzi dochází *před* vyvoláním uživatelské procedury pro zpracování zprávy. Z tohoto důvodu nesmí být konverze požadována na původním odesílacím kanálu. Pokud se referenční zpráva předává přes přechodný uzel, hromadná data se konvertují při odeslání z mezilehlého uzlu, je-li požadováno.
- Referenční zprávy nemohou být segmentovány.

Použití struktur MQRMH a MQMD

Viz MQRMH a MQMD pro popis polí v záhlaví referenční zprávy a v deskriptoru zpráv.

Ve struktuře MQMD nastavte pole *Format* na hodnotu MQFMT_REF_MSG_HEADER. Formát MQHREF je při požadavku na MQGET automaticky převeden produktem IBM MQ spolu s veškerými hromadným daty, která následují.

Dále je uveden příklad použití polí *DataLogicalOffset* a *DataLogicalLength* MQRMH:

Aplikace uvedení do provozu může obsahovat referenční zprávu obsahující:

- Žádná fyzická data
- *DataLogicalLength* = 0 (tato zpráva představuje celý objekt)
- *DataLogicalOffset* = 0.

Za předpokladu, že objekt má délku 70 000 bajtů, odesílající uživatelská procedura odešle prvních 40 000 bajtů v kanálu v referenční zprávě obsahující:

- 40 000 bajtů fyzických dat za MQRMH
- *DataLogicalLength* = 40000
- *DataLogicalOffset* = 0 (od začátku objektu).

Poté umístí jinou zprávu do přenosové fronty obsahující:

- Žádná fyzická data
- *DataLogicalLength* = 0 (až do konce objektu). Můžete zde zadat hodnotu 30 000.
- *DataLogicalOffset* = 40000 (od tohoto bodu).

Je-li tato uživatelská procedura pro odeslání zprávy zobrazena uživatelskou procedurou odeslání zprávy, je připojeno zbývajících 30 000 bajtů dat a pole jsou nastavena na následující hodnoty:

- 30 000 bajtů fyzických dat za MQRMH

- *DataLogicalLength* = 30000
- *DataLogicalOffset* = 40000 (od tohoto bodu).

Příznak MQRMHF_LAST je také nastaven.

Popis ukázkových programů poskytovaných pro použití referenčních zpráv naleznete v tématu “Použití ukázkových programů na více platformách” na stránce 1035.

Čekání na zprávy

Pokud chcete, aby program čekal na příchod zprávy do fronty, zadejte volbu MQGMO_WAIT v poli *Options* struktury MQGMO.

Použijte pole *WaitInterval* struktury MQGMO k určení maximální doba (v milisekundách), po kterou má volání MQGET čekat na příchod zprávy do fronty.

Pokud se zpráva nedostaví do této doby, volání MQGET se dokončí s kódem příčiny MQRC_NO_MSG_AVAILABLE.

Můžete zadat neomezený interval čekání pomocí konstanty MQWI_UNLIMITED v poli *WaitInterval* . Události mimo váš ovládací prvek však mohou způsobit čekání vašeho programu na delší dobu, takže tuto konstantu použijte opatrně. Aplikace produktu IMS nesmí určovat neomezený interval čekání, protože by se zabránilo ukončení systému IMS . (Když je IMS ukončeno, vyžaduje to, aby všechny závislé regiony byly ukončeny.) Místo toho mohou aplikace IMS určit konečný interval čekání; potom, pokud je volání dokončeno bez načtení zprávy po tomto intervalu, vydejte další volání MQGET s volbou čekání.

Poznámka: Pokud více než jeden program čeká ve stejné sdílené frontě na *odebrání* zprávy, aktivuje se příchozí zpráva pouze jeden program. Avšak pokud čeká na prohlížení zprávy více než jeden program, všechny programy lze aktivovat. Další informace naleznete v popisu pole *Options* struktury MQGMO v produktu MQGMO.

Pokud se stav fronty nebo správce front změní před vypršením čekací doby, dojde k následujícím akcím:

- Pokud správce front přejde do klidového stavu a vy jste použili volbu MQGMO_FAIL_IF_QUIESCING, čekání je zrušeno a volání MQGET se dokončí s kódem příčiny MQRC_Q_MGR_QUIESCING. Bez této volby zůstává volání čekání.
- z/OS Pokud v produktu z/OSvstoupí do klidového stavu připojení (pro aplikaci CICS nebo IMS) a vy jste použili volbu MQGMO_FAIL_IF_QUIESCING, čekání je zrušeno a volání MQGET se dokončí s kódem příčiny MQRC_CONN_QUIESCING. Bez této volby zůstává volání čekání.
- Je-li správce front nucen zastavit nebo je zrušen, je volání MQGET dokončeno buď s kódem příčiny MQRC_Q_MGR_STOPPING nebo MQRC_CONNECTION_BROKEN.
- Změní-li se atributy fronty (nebo fronty, na kterou se název fronty řeší), takže požadavky na získání jsou nyní blokovány, čekání je zrušeno a volání MQGET se dokončí s kódem příčiny MQRC_GET_INHIBITED.
- Změní-li se atributy fronty (nebo fronty, na kterou se název fronty řeší) takovým způsobem, že je vyžadována volba FORCE, bude čekání zrušeno a volání MQGET se dokončí s kódem příčiny MQRC_OBJECT_CHANGED.

z/OS Chcete-li, aby vaše aplikace čekala ve více než jedné frontě, použijte poskytovanou službu pro signál IBM MQ for z/OS (viz “signalizace” na stránce 764). Další informace o okolnostech, za kterých k těmto akcím dochází, najdete v tématu MQGMO.

signalizace

Signalizace je podporována pouze v systému IBM MQ for z/OS.

Signalizace je volba na volání MQGET, která umožňuje operačnímu systému upozornění (nebo *signál*) program, když je doručena očekávaná zpráva do fronty. To je jako funkce *get with wait* popsanou v tématu “Čekání na zprávy” na stránce 764 , protože umožňuje vašemu programu pokračovat s jinou prací při čekání na signál. Pokud však použijete signalizaci, můžete uvolnit aplikační podproces a spolehnout se na operační systém, aby program upozornili, až přijde zpráva.

Nastavení signálu

Chcete-li nastavit signál, proveďte následující ve struktuře MQGMO, kterou používáte ve volání MQGET:

1. Nastavte volbu MQGMO_SET_SIGNAL v poli *Options* .
2. Nastavte maximální životnost signálu v poli *WaitInterval* . Tato hodnota určuje dobu (v milisekundách), po kterou má produkt IBM MQ monitorovat frontu. Použijte hodnotu MQWI_UNLIMITED k určení neomezeného života.

   Poznámka: Aplikace produktu IMS nesmí určovat neomezený interval čekání, protože tento proces zabrání ukončení systému IMS . (Když je IMS ukončeno, vyžaduje to, aby všechny závislé regiony byly ukončeny.) Místo toho mohou aplikace produktu IMS kontrolovat stav ECB v pravidelných intervalech (viz krok 3). Program může mít ve stejnou dobu signály nastavené na několika manipulátorů fronty:
3. Do pole *Signal1* zadejte adresu modulu *Event Control Block* (ECB). To vás upozorní na výsledek vašeho signálu. Uskladnění ECB musí zůstat k dispozici, dokud nebude fronta uzavřena.

Poznámka: Volbu MQGMO_SET_SIGNAL nelze použít s volbou MQGMO_WAIT.

Kdy přijde zpráva

Je-li doručena vhodná zpráva, je ECB vrácena kód dokončení.

Kód dokončení popisuje jednu z následujících možností:

- Zpráva, pro kterou jste nastavili signál, byla doručena do fronty. Zpráva není vyhrazena pro program, který požadoval signál, takže program musí znovu zadat volání MQGET, aby se zpráva dostala.

  Poznámka: Jiná aplikace by mohla obdržet zprávu v době mezi přijetím signálu a vydáním jiného volání MQGET.
- Interval čekání, který jste nastavili, má vypršenou platnost a zpráva, pro kterou jste nastavili signál, nebyla doručena do fronty. IBM MQ zrušil signál.
- Signál byl zrušen. K tomu dojde například v případě, že je správce front zastaven, nebo dojde ke změně atributu fronty, takže volání MQGET již nejsou povolena.

Je-li již ve frontě vhodná zpráva, volání MQGET se dokončí stejným způsobem jako volání MQGET bez signalizace. Také je-li detekována chyba okamžitě, volání se dokončí a návratové kódy jsou nastaveny.

Když je volání přijato a žádná zpráva není okamžitě k dispozici, řízení je vráceno do programu, aby mohl pokračovat s jinou prací. Nejsou nastaveny žádné výstupní pole v deskriptoru zpráv, ale parametr **CompCode** je nastaven na hodnotu MQCC_WARNING a parametr **Reason** je nastaven na hodnotu MQRC_SIGNAL_REQUEST_ACCEPTED.

Informace o tom, co produkt IBM MQ může vrátit do vaší aplikace, když provádí volání MQGET pomocí signalizace, najdete v tématu MQGET.

Pokud program nemá jinou práci na práci, zatímco čeká na odeslání ECB, může ECB čekat na to, že bude používat:

- Pro program CICS Transaction Server pro z/OS , příkaz EXEC CICS WAIT EXTERNAL
- Pro dávkové programy a programy IMS je makro WAIT z/OS

Pokud se stav fronty nebo správce front změní, zatímco je signál nastaven (to znamená, že ECB dosud nebyla zveřejněna), provedou se následující akce:

- Pokud správce front přejde do klidového stavu a vy jste použili volbu MQGMO_FAIL_IF_QUIESCING, je tento signál zrušen. ECB je uveřejněna s kódem dokončení MQEC_Q_MGR_QUIESCING. Bez této volby zůstává signál nastaven.
- Je-li správce front nucen přestat nebo je zrušen, signál je zrušen. Signál je dodáván s kódem dokončení MQEC_WAIT_CANCELED.
- Změní-li se atributy fronty (nebo fronty, na kterou se název fronty řeší) tak, že požadavky na získání jsou nyní blokovány, je signál zrušen. Signál je dodáván s kódem dokončení MQEC_WAIT_CANCELED.

Poznámka:

1. Pokud více než jeden program nastavila signál ve stejné sdílené frontě k odstranění zprávy, aktivuje se příchozí zpráva pouze v jednom programu. Avšak pokud čeká na prohlížení zprávy více než jeden program, všechny programy lze aktivovat. Pravidla, která správce front postupuje při rozhodování o tom, které aplikace mají být aktivovány, jsou stejné jako u čekacích aplikací: další informace naleznete v popisu pole *Options* struktury MQGMO ve volbě MQGMO-Get-message options.
2. Pokud existuje více než jedna volání MQGET čekající na stejnou zprávu se směsí voleb čekání a signálu, každá čekající volání se považuje za stejnou hodnotu. Další informace naleznete v popisu pole *Options* struktury MQGMO ve struktuře MQGMO-Get-message options.
3. Za určitých podmínek je možné, aby volání MQGET znovu načetla zprávu a aby byl odeslán signál (výsledkem je příchod stejné zprávy). To znamená, že když váš program vydá další volání MQGET (protože byl doručen signál), nemůže být dostupná žádná zpráva. Navrhněte svůj program pro testování této situace.

Informace o tom, jak nastavit signál, najdete v popisu volby MQGMO_SET_SIGNAL a v poli *Signal1* v Signal1.

Vynechání odvolání

Aplikační program můžete zabránit v zadání smyčky *MQGET-error-backout* zadáním volby **MQGMO_MARK_SKIP_BACKOUT** na volání MQGET.

Poznámka: Podporováno pouze na IBM MQ for z/OS.

Jako součást pracovní jednotky může aplikační program vydat jeden nebo více volání MQGET k získání zpráv z fronty. Pokud aplikační program zjistí chybu, může vrátit jednotku práce zpět. Tím se obnoví všechny prostředky aktualizované během této jednotky práce do stavu, ve kterém se nacházely před spuštěním jednotky práce, a znovu obnoví zprávy načtené voláním MQGET.

Po opětovném zavedení jsou tyto zprávy k dispozici pro následující volání MQGET vydaná aplikačním programem. V mnoha případech to nezpůsobí problém pro aplikační program. Avšak v případech, kdy nelze obcházet chybu vedoucí k vrácení zpět, může mít zpráva ve frontě znovu zavedena zpráva, která může způsobit, že aplikační program vstoupí do smyčky *MQGET-error-backout* .

Chcete-li tomuto problému předejít, zadejte v rámci příkazu MQGET volbu MQGMO_MARK_Send_KIP_BACOUT. Tím se označí požadavek MQGET jako nezahrnutý do odvolání zahájeného aplikací; to znamená, že nesmí být vrácena. Použití této volby znamená, že když dojde k odvolání, aktualizace z jiných prostředků se zálohují podle potřeby, ale s označenou zprávou se zachází, jako by byla načtena pod novou pracovní jednotkou.

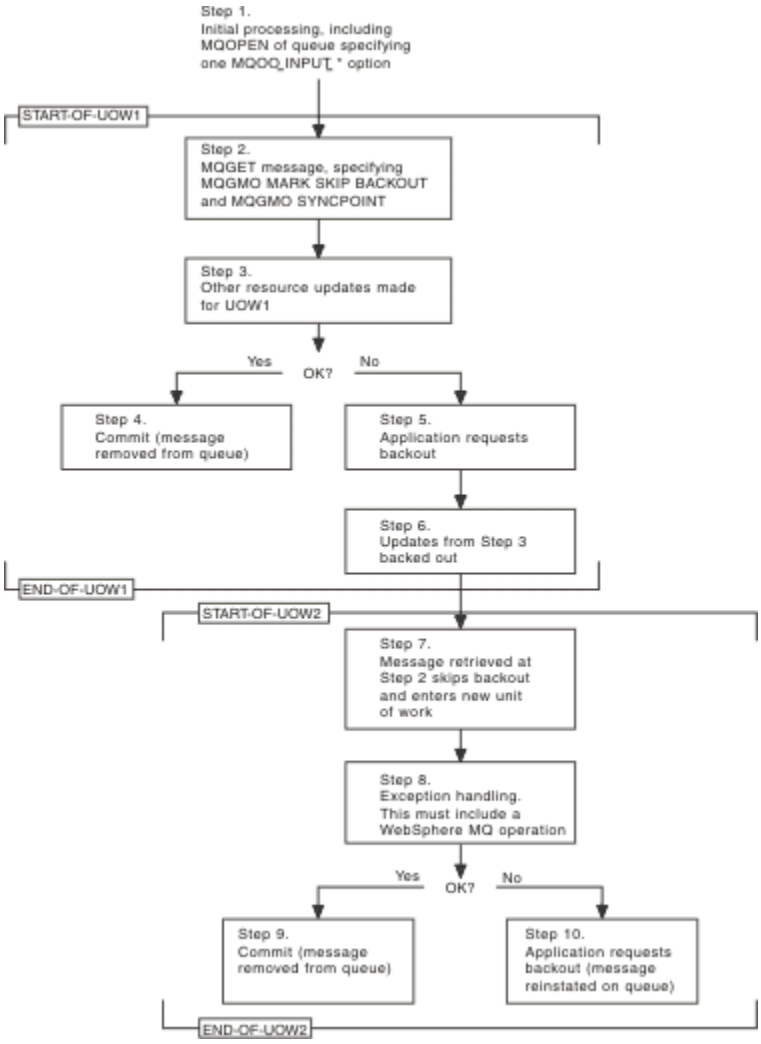
Aplikační program musí zadat výzvu IBM MQ buď k potvrzení nové pracovní jednotky, nebo k odvrácení nové pracovní jednotky. Program může například provádět zpracování výjimek, jako je například informování odesílatele o tom, že zpráva byla zahozena, a potvrdit jednotku práce tak, aby byla odstraněna zpráva z fronty, pokud je nová pracovní jednotka vrácena (z jakéhokoli důvodu) zpráva byla znovu zavedena do fronty.

V rámci pracovní jednotky může existovat pouze jeden požadavek MQGET označený jako přeskočení odvolání. Může však existovat několik dalších zpráv, které nejsou označeny jako přeskočení z odvolání. Jakmile je zpráva označena jako přeskočena, všechna další volání MQGET v rámci pracovní jednotky, která určuje MQGMO_MARK_SKIP_BACKUP, selžou s kódem příčiny MQRC_SECOND_MARK_NOT_ALLOWED.

Poznámka:

1. Označená zpráva se přeskočí pouze v případě, že je jednotka práce, která ji obsahuje, ukončena požadavkem aplikace na jeho vrácení. Je-li jednotka práce vrácena z jiných příčin, je zpráva vrácena do fronty stejným způsobem, jako by byla, pokud nebyla označena, aby přeskočila odvolání.
2. Vynechat vrácení není podporováno v rámci uložených procedur produktu Db2 , které se podílí na jednotkách práce řízených RRS. Například volání MQGET s volbou MQGMO_MARK_KIP_BACKOUT selže s kódem příčiny MQRC_OPTION_ENVIRONMENT_ERROR.

Obrázek 73 na stránce 767 znázorňuje typickou posloupnost kroků, které může aplikační program obsahovat, je-li požadován požadavek MQGET pro přeskočení odvolání.



Obrázek 73. Vynechání odvolání pomocí funkce MQGMO_MARK_SKIP_BACKUP

Kroky v produktu Obrázek 73 na stránce 767 jsou:

Krok 1
K počátečnímu zpracování dojde v rámci transakce, včetně volání MQOPEN pro otevření fronty (zadáním jedné z voleb MQOO_INPUT_ *, aby bylo možné získat zprávy z fronty v kroku 2).

Krok 2
Je volána funkce MQGET s hodnotami MQGMO_SYNCPOINT a MQGMO_MARK_SKIP_BACKUP. Funkce MQGMO_SYNCPOINT je vyžadována, protože operace MQGET musí být v rámci pracovní jednotky pro funkci MQGMO_MARK_SKIP_BAC_BE platnou. V Obrázek 73 na stránce 767 je tato jednotka práce odkazována jako UOW1.

Krok 3
Další aktualizace prostředků se provádějí v části UOW1. Ty mohou zahrnovat další volání MQGET (vydaná bez MQGMO_MARK_SKIP_BACKUUT).

Krok 4
Všechny aktualizace z kroků 2 a 3 jsou dokončeny podle potřeby. Aplikační program potvrdí aktualizace a UOW1 skončí. Zpráva načtená z kroku 2 bude odebrána z fronty.

Krok 5
Některé z aktualizací z kroků 2 a 3 nejsou dokončeny podle potřeby. Aplikační program požaduje, aby aktualizace provedené během těchto kroků byly zálohovány.

Krok 6
Aktualizace provedené v kroku 3 jsou vráceny zpět.

Krok 7
Požadavek MQGET, který byl proveden v kroku 2, přeskočí a stane se součástí nové pracovní jednotky, UOW2.

Krok 8
UOW2 provádí zpracování výjimek v reakci na objekt UOW1 , který má být zálohován. (Například volání MQPUT do jiné fronty, označující, že došlo k problému, který způsobil, že byl UOW1 zálohován.)

Krok 9
Krok 8 se dokončí podle potřeby, aplikační program potvrdí aktivitu a ukončí se UOW2 . Vzhledem k tomu, že požadavek MQGET je součástí UOW2 (viz krok 7), způsobí toto potvrzení, že zpráva bude odebrána z fronty.

Krok 10
Krok 8 není dokončen, jak je požadováno, a aplikační program zálohuje UOW2. Protože požadavek na získání zprávy je součástí UOW2 (viz krok 7), je také zálohován a znovu vrácen do fronty. Nyní je k dispozici pro další volání MQGET vydaná tímto nebo jiným aplikačním programem (ve stejné podobě jako jakákoli jiná zpráva ve frontě).

Převod dat aplikace

Je-li to nezbytné, MCAs převádí deskriptor zprávy a data záhlaví do požadované znakové sady a kódování. Převod může provést buď konec odkazu (tj. lokální agent MCA nebo vzdálená MCA), může tento spoj provést.

Když aplikace vkládá zprávy do fronty, lokální správce front přidá řídicí informace do deskriptorů zpráv, aby usnadnil řízení zpráv, když jsou zpracovány správci front a MCAs. V závislosti na daném prostředí jsou datová pole záhlaví zprávy vytvořena ve znakové sadě a v kódování lokálního systému.

Když přesouváte zprávy mezi systémy, někdy je třeba převést data aplikace do znakové sady a kódování, které je vyžadováno přijímajícím systémem. To lze provést buď z aplikačních programů na přijímajícím systému, nebo pomocí MCA na odesílajícím systému. Je-li konverze dat podporována na přijímajícím systému, použijte aplikační programy k převedení dat aplikace, spíše než v závislosti na konverzi, která se již vyskytla na odesílajícím systému.

Data aplikace jsou převedena v rámci aplikačního programu, když určujete volbu MQGMO_CONVERT v poli *Options* struktury MQGMO předané do volání MQGET a *all* následující příkazy jsou pravdivé:

- Pole *CodedCharSetId* nebo *Encoding* nastavená ve struktuře MQMD přidružené ke zprávě ve frontě se liší od polí *CodedCharSetId* nebo *Encoding* nastavených ve struktuře MQMD určené v rámci volání MQGET.
- Pole *Format* ve struktuře MQMD přidružené ke zprávě není MQFMT_NONE.
- Hodnota *BufferLength* zadaná v rámci volání MQGET není nulová.
- Délka dat zprávy není nula.
- Správce front podporuje převod mezi poli *CodedCharSetId* a *Encoding* zadanými ve strukturách MQMD přidruženému ke zprávě a voláním MQGET. Viz CodedCharSetId a Encoding , kde jsou uvedeny podrobnosti o identifikátorech kódované znakové sady a podporovaných kódování počítače.
- Správce front podporuje převod formátu zpráv. Je-li pole *Format* struktury MQMD přidružené ke zprávě jedním z vestavěných formátů, může správce front zprávu převést. Pokud *Format* není jedním z vestavěných formátů, musíte napsat uživatelskou proceduru pro převod dat, abyste tuto zprávu převedli.

Pokud má odesílající agent MCA převést data, zadejte klíčové slovo CONVERT (YES) na definici každého odesílatele nebo kanálu serveru, pro který je konverze vyžadována. Pokud dojde k selhání konverze dat, zpráva se odešle do fronty DLQ v odesílajícím správci front a pole *Feedback* struktury MQDLH označuje důvod. Pokud nelze zprávu vložit do fronty nedoručených zpráv, kanál se zavře a nepřevedená zpráva zůstane v přenosové frontě. Data conversion within applications rather than at sending MCAs eliminate this situation.

Jako pravidlo jsou data ve zprávě, která jsou popsána jako *znaková* data pomocí vestavěného formátu nebo uživatelské procedury pro převod dat, převedena z kódované znakové sady použité ve zprávě na požadovanou a *číselná* pole jsou převedena na požadované kódování.

Další podrobnosti o konvencích zpracování konverze použitých při převádění vestavěných formátů a na informace o psaní vlastních uživatelských procedur pro převod dat naleznete v tématu “Zápis uživatelských procedur pro převod dat” na stránce 948. Viz také Národní jazyky a Kódování počítače , kde najdete informace o tabulkách jazykové podpory a o podporovaných kódování počítače.

Převod znaků nového řádku EBCDIC

Potřebujete-li zajistit, aby data odesílaná z platformy EBCDIC do ASCII jedna byla identická s daty, která obdržíte znovu, musíte řídit konverzi znaků znaku EBCDIC nového řádku.

Můžete to provést pomocí přepínače závislého na platformě, která nutí IBM MQ k použití nezměněných převodních tabulek, ale musíte si být vědomi nekonzistentního chování, které může mít za následek.

Problém vzniká, protože znak nového řádku EBCDIC se nepřevádí konzistentně přes platformy nebo konverzní tabulky. Výsledkem je, že jsou-li data zobrazena na platformě ASCII, může být formátování nesprávné. Tím by bylo obtížné například vzdáleně spravovat systém IBM i z platformy ASCII pomocí RUNMQSC.

Další informace o konverzi dat ve formátu EBCDIC do formátu ASCII najdete v tématu Převod dat .

Procházení zpráv ve frontě

Tyto informace použijte k vyhledání informací o procházení zpráv ve frontě pomocí volání MQGET.

Chcete-li použít volání MQGET k procházení zpráv ve frontě, postupujte takto:

1. Chcete-li otevřít frontu pro procházení určením volby MQOO_BROWSE, volejte MQOPEN.
2. Chcete-li procházet první zprávu ve frontě, zavolejte příkaz MQGET s volbou MQGMOPRE_FIRST. Chcete-li vyhledat požadovanou zprávu, opakovaně volejte MQGET s volbou MQGMO_BROTE_NEXT, abyste prošli mnoha zprávami.

   Chcete-li zobrazit všechny zprávy, je třeba nastavit pole *MsgId* a *CorrelId* struktury MQMD na hodnotu null po každém volání MQGET.
3. Chcete-li zavřít frontu, volejte příkaz MQCLOSE.

Kurzor procházení
Když otevírejte frontu pro procházení (MQOPEN), volání založí kurzor procházení pro použití s voláními MQGET, které používají jednu z voleb procházení. Můžete si myslet na kurzor procházení jako na logický ukazatel, který je umístěn před první zprávou ve frontě.

Můžete mít více než jeden kurzor procházení (z jednoho programu) tak, že zadáte několik požadavků MQOPEN pro stejnou frontu.

Když voláte MQGET pro procházení, použijte jednu z následujících možností ve struktuře MQGMO:

NEJPRVE MQGMO_BROWSE_FIRST
Získá kopii první zprávy, která splňuje podmínky určené ve struktuře MQMD.

PŘÍŠTĚ MQGMO_BROWSE_NEXT
Získá kopii další zprávy, která splňuje podmínky určené ve struktuře MQMD.

MQGMO_BROWSE_MSG_UNDER_CURSOR
Získá kopii zprávy, na kterou se aktuálně odkazuje kurzor, tj. ten, který byl naposledy načten pomocí volby MQGMO_BROFIRST FIRST nebo MQGMO_BRONEXT.

Ve všech případech zůstává zpráva ve frontě.

Když otevřete frontu, kurzor procházení je umístěn logicky těsně před první zprávou ve frontě. To znamená, že pokud provedete volání MQGET okamžitě po volání MQOPEN, můžete pomocí volby MQGMO_BROWSE NEXT procházet první zprávu; nemusíte používat volbu MQGMO_BROTED FIRST.

Pořadí, ve kterém se zprávy kopírují z fronty, je určeno atributem **MsgDeliverySequence** fronty. (Další informace viz “Pořadí, ve kterém jsou zprávy načítány z fronty” na stránce 742.)

- “Fronty v posloupnosti FIFO (první dovnitř, první ven)” na stránce 770
- “Fronty v posloupnosti priorit” na stránce 770

- “Nepotvrzené zprávy” na stránce 770
- “Změnit na pořadí fronty” na stránce 770
- “Použití indexu fronty” na stránce 770

Fronty v posloupnosti FIFO (první dovnitř, první ven)

První zpráva ve frontě v tomto pořadí je zpráva, která byla na frontě nejdelší.

Chcete-li číst zprávy sekvenčně ve frontě, použijte příkaz MQGMOROWSE_NEXT. Zobrazí se zprávy vložené do fronty během procházení, protože fronta v této posloupnosti má zprávy umístěné na konci. Když kurzor rozpozná, že dosáhla konce fronty, kurzor procházení zůstane tam, kde je a vrátí se s MQRC_NO_MSG_AVAILABLE. Můžete ji pak buď nechat čekat na další zprávy, nebo ji resetovat na začátek fronty s voláním MQGMOPRADE_FIRST.

Fronty v posloupnosti priorit

První zpráva ve frontě v tomto pořadí je zpráva, která byla ve frontě nejdelší a která má nejvyšší prioritu v době, kdy bylo vydáno volání MQOPEN.

Chcete-li číst zprávy ve frontě, použijte příkaz MQGMO_BROE_NEXT.

Kurzor procházení ukazuje na další zprávu, která pracuje od priority první zprávy, která má být dokončena se zprávou na nejnižší prioritě. Během této doby prochází všechny zprávy, které byly do fronty vloženy, pokud mají prioritu stejnou nebo nižší, než je zpráva identifikovaná aktuálním kurzorem procházení.

Všechny zprávy zařazené do fronty s vyšší prioritou lze procházet pouze následujícími způsoby:

- Otevřením fronty znovu procházet, v tomto okamžiku je vytvořen nový kurzor procházení
- Použití volby MQGMOPRE_FIRST

Nepotvrzené zprávy

Nepotvrzená zpráva není nikdy viditelná pro procházení; kurzor procházení přeskočí přes něj.

Zprávy v rámci pracovní jednotky nelze procházet, dokud není potvrzena jednotka práce. Zprávy neměňte jejich umístění ve frontě, takže přeskočené, nepotvrzené zprávy nebudou vidět, i když jsou *jsou* potvrzeny, pokud nepoužijete volbu MQGMOPRE_FIRST a znovu pracují s frontou.

Změnit na pořadí fronty

Je-li pořadí doručení zprávy změněno z priority na FIFO, zatímco ve frontě jsou zprávy, pořadí zpráv, které jsou již ve frontě, se nezmění. Zprávy přidané do fronty později, mají výchozí prioritu fronty.

Použití indexu fronty

Při procházení indexované fronty obsahující pouze zprávy s jednou prioritou (buď perzistentní, nebo přechodná, nebo obě) používá správce front index k procházení při použití určitých forem procházení.

Poznámka: Podporováno pouze na IBM MQ for z/OS.

Pokud indexovaná fronta obsahuje pouze zprávy s jednou prioritou, používají se některé z následujících forem procházení:

1. Pokud je fronta indexována podle MSGID, požadavky na procházení, které předají MSGID ve struktuře MQMD, se zpracují s použitím indexu k nalezení cílové zprávy.
2. Pokud je fronta indexována podle CORRELID, procházejí požadavky, které projdou CORRELID ve struktuře MQMD, za použití indexu k nalezení cílové zprávy.
3. Pokud je fronta indexována pomocí GROUPID, procházejí požadavky, které předáte GROUPID ve struktuře MQMD, zpracované s použitím indexu k nalezení cílové zprávy.

Pokud požadavek na procházení nepředává ID MSGID, CORRELID nebo GROUPID ve struktuře MQMD, fronta je indexována a je vrácena zpráva, musí být nalezena položka indexu pro tuto zprávu a informace v ní použité k aktualizaci kurzoru pro procházení. Použijete-li široký výběr hodnot indexu, nepřidává se do požadavku na procházení významné další zpracování.

Procházení zpráv, pokud je délka zprávy neznámá

Chcete-li procházet zprávu, když neznáte velikost zprávy a nechcete použít pole *MsgId*, *CorrelId*nebo *GroupId* k vyhledání zprávy, můžete použít volbu MQGMOROWS_MSG_UNDER_CURSOR:

1. Zadejte příkaz MQGET s:
   - buď volba MQGMO_BROWSE_FIRST nebo MQGMO_BRONEXT NEXT
   - Volba MQGMO_ACCEPT_TRUNCATED_MSG
   - Nula vyrovnávací paměti

   Poznámka: Pokud je pravděpodobné, že jiný program získá stejnou zprávu, zvažte použití volby MQGMO_LOCK. Je třeba vrátit MQRC_TRUNCATED_MSG_ACCEPTED.
2. Chcete-li přidělit potřebnou paměť, použijte navrácenou hodnotu *DataLength* .
3. Zadejte příkaz MQGET s parametrem MQGMO_BROWS_MSG_UNDER_CURSOR.

Zpráva ukazovala na načtenou poslední, která byla načtena; kurzor procházení se nepřesunul. Můžete zvolit buď zamknutí zprávy pomocí volby MQGMO_LOCK, nebo odemknout zamčenou zprávu pomocí volby MQGMO_UNLOCK.

Volání selže, pokud nebyla úspěšně vydána žádná operace MQGET s volbami MQGMO_BROWSE_FIRST nebo MQGMO_BOTE NEXT od té doby, kdy byla fronta otevřena.

Odebrání zprávy, kterou jste procházeli
Z fronty můžete odebrat zprávu, kterou jste již prohlédl, za předpokladu, že jste frontu otevřeli pro odebrání zpráv a také pro procházení. (Je třeba určit jednu z voleb MQOO_INPUT_ * a také volbu MQOO_BROWSE v rámci volání MQOPEN.)

Chcete-li zprávu odebrat znovu, obraťte se na příkaz MQGET znovu, ale v poli *Options* struktury MQGMO určete MQGMO_MSG_UNDER_CURSOR.V takovém případě volání MQGET ignoruje pole *MsgId*, *CorrelId*a *GroupId* struktury MQMD.

V době mezi jednotlivými kroky při procházení a odebrání může jiný program odebírat zprávy z fronty, včetně zprávy pod kurzorem procházení. V takovém případě volání MQGET vrátí kód příčiny, který říká, že zpráva není k dispozici.

Procházení zpráv v logickém pořadí

“Logické a fyzické uspořádání” na stránce 743 vysvětluje rozdíl mezi logickým a fyzickým pořadím zpráv ve frontě. Tento rozdíl je zvláště důležitý při procházení fronty, protože obecně se zprávy neodstraňují a operace procházení nemusí nutně začínat na začátku fronty.

Pokud některá aplikace prochází různými zprávami jedné skupiny (s použitím logického pořadí), je důležité, aby se za účelem dosažení začátku další skupiny mělo následovat logické pořadí, protože poslední zpráva jedné skupiny se může stát fyzicky *po* první zprávě další skupiny. Volba MQGMO_LOGICAL_ORDER zajišťuje, aby při skenování fronty bylo dodržováno logické pořadí.

Použijte funkci MQGMO_ALL_MSGS_AVAILABLE (nebo MQGMO_ALL_SEGMENTS_AVAILABLE) s péčí o operace procházení. Zvažte případ logických zpráv s MQGMO_ALL_MSGS_AVAILABLE. Výsledkem je, že logická zpráva je k dispozici pouze v případě, že jsou přítomny také všechny zbývající zprávy ve skupině. Pokud tomu tak není, zpráva se předá znovu. To může znamenat, že když se chybějící zprávy dostaví později, nejsou při procházení zaznamenány operací procházení.

Jsou-li například přítomny následující logické zprávy,

```
Logical message 1 (not last) of group 123
```

```
   Logical message 1 (not last) of group 456
   Logical message 2 (last)     of group 456
```

a funkce procházení je vydána s parametrem MQGMO_ALL_MSGS_AVAILABLE, je vrácena první logická zpráva skupiny 456 a ponechá se kurzor procházení této logické zprávy. Pokud je nyní doručena druhá (poslední) zpráva skupiny 123:

```
   Logical message 1 (not last) of group 123
   Logical message 2 (last)     of group 123
   Logical message 1 (not last) of group 456  <=== browse cursor
   Logical message 2 (last)     of group 456
```

a je vydána stejná funkce browse-next, nevšiml si, že skupina 123 je nyní dokončena, protože první zpráva této skupiny je *před* kurzorem procházení.

V některých případech (například, pokud jsou zprávy načítány destruktivně, když je skupina přítomna v celém rozsahu), můžete použít MQGMO_ALL_MSGS_AVAILABLE spolu s MQGMO_BROFIRST. Jinak musíte zopakovat procházení procházení, aby bylo možné vzít na vědomí nově přijaté zprávy, které byly vynechány; pouze zadání MQGMO_WAIT spolu s MQGMO_BE NEXT a MQGMO_ALL_MSGS_AVAILABLE nebude brát v úvahu jejich počet. (To se také děje s vysoce prioritními zprávami, které mohou přicházet po skenování zpráv, je dokončeno.)

Další sekce se zabývají procházení příkladů, které se zabývají nesegmentovanými zprávami; segmentované zprávy dodržují podobné zásady.

Procházení zpráv ve skupinách
V tomto příkladu prochází aplikace přes každou zprávu ve frontě v logickém pořadí.

Zprávy ve frontě mohou být seskupeny. Pro seskupené zprávy aplikace nechce spustit zpracování žádné skupiny, dokud nedorazily všechny zprávy v ní obsažené. Funkce MQGMO_ALL_MSGS_AVAILABLE je proto určena pro první zprávu ve skupině; pro následující zprávy ve skupině je tato volba zbytečná.

Funkce MQGMO_WAIT se v tomto příkladu používá. Přestože však lze čekat na doručení nové skupiny z důvodů v produktu “Procházení zpráv v logickém pořadí” na stránce 771, není splněna, pokud kurzor procházení již prošel první logickou zprávou ve skupině, a zbývající zprávy jsou nyní doručeny. Nicméně čekání na vhodný interval zajišťuje, že aplikace nebude při čekání na nové zprávy nebo segmenty neustále smyčkat.

Funkce MQGMO_LOGICAL_ORDER se používá v celém rozsahu, aby bylo zaručeno, že skenování je v logickém pořadí. Tento rozdíl je v rozporu s destruktivním parametrem MQGET, kde je odebíraná každá skupina, MQGMO_LOGICAL_ORDER, při hledání první (nebo jediné) zprávy ve skupině.

Předpokládá se, že vyrovnávací paměť aplikace je vždy dostatečně velká, aby udržela celou zprávu, ať už byla zpráva segmentována, či nikoli. MQGMO_COMPLETE_MSG je proto určen pro každý příkaz MQGET.

Níže je uveden příklad procházení logických zpráv ve skupině:

```
/* Browse the first message in a group, or a message not in a group */
GMO.Options = MQGMO_BROWSE_NEXT | MQGMO_COMPLETE_MSG | MQGMO_LOGICAL_ORDER
| MQGMO_ALL_MSGS_AVAILABLE | MQGMO_WAIT
MQGET GMO.MatchOptions = MQMO_MATCH_MSG_SEQ_NUMBER, MD.MsgSeqNumber = 1
/* Examine first or only message */
...

GMO.Options = MQGMO_BROWSE_NEXT | MQGMO_COMPLETE_MSG | MQGMO_LOGICAL_ORDER
do while ( GroupStatus == MQGS_MSG_IN_GROUP )
   MQGET
   /* Examine each remaining message in the group */
   ...
```

Skupina se opakuje, dokud není navrácena hodnota MQRC_NO_MSG_AVAILABLE.

Procházení a načítání destruktivně
V tomto příkladu aplikace prochází každou z logických zpráv v rámci skupiny před tím, než se rozhodne, zda tuto skupinu destruktivně načíst.

První část tohoto příkladu je podobná předchozí části. Avšak v tomto případě, když jsme procházeli celou skupinou, rozhodli jsme se vrátit a získat ji destruktivním způsobem.

Protože každá skupina je v tomto příkladu odstraněna, MQGMO_LOGICAL_ORDER se nepoužije při hledání první nebo jediné zprávy ve skupině.

Zde je uveden příklad procházení a následné načtení destruktivně:

```
GMO.Options = MQGMO_BROWSE_NEXT | MQGMO_COMPLETE_MSG | MQGMO_LOGICAL_ORDER
            | MQGMO_ALL_MESSAGES_AVAILABE | MQGMO_WAIT
do while ( GroupStatus == MQGS_MSG_IN_GROUP )
   MQGET
   /* Examine each remaining message in the group (or as many as
      necessary to decide whether to get it destructively) */
   ...

if ( we want to retrieve the group destructively )

   if ( GroupStatus == ' ' )
      /* We retrieved an ungrouped message */
      GMO.Options = MQGMO_MSG_UNDER_CURSOR | MQGMO_SYNCPOINT
      MQGET GMO.MatchOptions = 0
      /* Process the message */
      ...

   else
      /* We retrieved one or more messages in a group.  The browse cursor */
      /* will not normally be still on the first in the group, so we have */
      /* to match on the GroupId and MsgSeqNumber = 1.                    */
      /* Another way, which works for both grouped and ungrouped messages,*/
      /* would be to remember the MsgId of the first message when it was  */
      /* browsed, and match on that.                                      */
      GMO.Options = MQGMO_COMPLETE_MSG | MQGMO_SYNCPOINT
      MQGET GMO.MatchOptions = MQMO_MATCH_GROUP_ID
                             | MQMO_MATCH_MSG_SEQ_NUMBER,
           (MQMD.GroupId      = value already in the MD)
            MQMD.MsgSeqNumber = 1
      /* Process first or only message */
      ...

      GMO.Options = MQGMO_COMPLETE_MSG | MQGMO_SYNCPOINT
                  | MQGMO_LOGICAL_ORDER
      do while ( GroupStatus == MQGS_MSG_IN_GROUP )
         MQGET
         /* Process each remaining message in the group */
...
```

Vyvarování se opakovaného doručení procházených zpráv
Pomocí určitých voleb otevření a voleb pro získání zprávy můžete označit zprávy jako procházené tak, aby nebyly znovu načteny aktuální nebo jinými spolupracujícími aplikacemi. Zprávy mohou být explicitně nebo automaticky označeny jako nové, aby je bylo možné znovu zpřístupnit pro prohlížení.

Pokud procházíte zprávy ve frontě, můžete je načíst v jiném pořadí, než je pořadí, ve kterém byste je získali, pokud jste je získali destruktivně. Zejména můžete několikrát procházet stejnou zprávu, která není možná, pokud byla odebrána z fronty. Chcete-li se této zprávě vyhnout, můžete *označit* zprávy při jejich prohlížení a zabránit načítání označených zpráv. Toto je někdy označováno jako *procházení s označením*. Chcete-li označit procházené zprávy, použijte volbu get message MQGMO_MARKWSE_HANDLE a k načtení pouze zpráv, které nejsou označeny, použijte MQGMO_UNMARKED_BERWSE_MSG. Pokud použijete kombinaci voleb MQGMO_BROFIRST FIRST, MQGMO_UNMARKED_BERWSE_MSG a MQGMO_MARK_BROWSE_HANDLE a zadáte opakované volání MQGET, budete každou zprávu ve frontě postupně načítán. Tím se zabrání opakovanému doručení zpráv i v případě, že operace MQGMO_Bated FIRST se používá k zajištění toho, že zprávy nebudou vynechány. Tato kombinace voleb může být reprezentována jedinou konstantou MQGMO_BROWSE_HANDLE. Pokud ve frontě nejsou žádné zprávy, které nebyly procházeny, je vrácen objekt MQRC_NO_MSG_AVAILABLE.

Pokud prohledávaní více aplikací prochází stejnou frontu, mohou otevřít frontu s volbami MQOO_CO_OP a MQOO_BROWSE. Manipulátor objektu vrácený jednotlivými MQOPEN je považován za součást spolupracující skupiny. Jakákoli zpráva vrácená voláním MQGET s uvedením volby MQGMO_MARK_BROWSE_CO_OP je považována za označenou pro tuto spolupracující sadu manipulátorů.

Je-li zpráva již nějakou dobu označena, může ji správce front automaticky zrušit a zpřístupnit k jeho opětovnému prohlížení. Atribut správce front MsgMarkBrowseInterval udává dobu v milisekundách, po kterou má zpráva zůstat označena pro spolupracující sadu manipulátorů. MsgMarkBrowseInterval z -1 znamená, že zprávy nejsou nikdy automaticky neoznačené.

Když se zastaví jeden proces nebo sada procesů pro spolupráci značení zpráv, všechny označené zprávy budou neoznačeny.

Příklady spolupráce při procházení

Můžete spustit více kopií aplikace dispečera k procházení zpráv ve frontě a zahájení odběratele na základě obsahu každé zprávy. V každém dispečeru otevřete frontu s MQOO_CO_OP. To znamená, že dispečery spolupracují a budou si být vědomi, že se jedná o označené zprávy. Každý dispečer poté provádí opakované volání MQGET, přičemž určuje volby MQGMO_BROTE_FIRST, MQGMO_UNMARKED_BROWSE_MSG a MQGMO_MARK_BROWSE_CO_OP (můžete použít jedinou konstantu MQGMO_BROWSE_CO_OP, která představuje tuto kombinaci voleb). Každá dispečerské aplikace pak načte pouze ty zprávy, které ještě nebyly označeny jinými spolupracujícími dispečery. Dispečer inicializuje spotřebitele a předává prvek MsgToken vrácený procesem MQGET pro spotřebitele, který destruktivně získá zprávu z fronty. Pokud spotřebitel zazálohuje příkaz MQGET zprávy, je zpráva k dispozici pro jeden z prohlížečů k opětnému odeslání, protože již není označena. Pokud spotřebitel neprovede příkaz MQGET ve zprávě, poté, co byl předán objekt MsgMarkBrowseInterval , správce front zruší označení zprávy pro spolupracující sadu manipulátorů a lze ji znovu odbavit.

Spíše než více kopií stejné aplikace dispečeru, můžete mít několik různých aplikací dispečera prohledávajících frontu, z nichž každá je vhodná pro zpracování podmnožiny zpráv ve frontě. V každém dispečeru otevřete frontu s MQOO_CO_OP. To znamená, že dispečery spolupracují a budou si být vědomi, že se jedná o označené zprávy.

- Je-li pro jeden dispečer důležité pořadí zpracování zpráv, každý dispečer provádí opakované volání MQGET, přičemž určuje volby MQGMO_BROTE_FIRST, MQGMO_UNMARKED_BROWSE_MSG a MQGMO_MARK_BROWSE_HANDLE (nebo MQGMO_BROWSE_HANDLE). Je-li procházená zpráva vhodná pro tento dispečer ke zpracování, potom provede volání MQGET s parametrem MQMO_MATCH_MSG_TOKEN, MQGMO_MARK_BROWSE_CO_OP a MsgToken vráceným předchozím voláním MQGET. Je-li volání úspěšné, dispečer inicializuje spotřebitele a předá mu MsgToken .
- Není-li pořadí zpracování zpráv důležité a očekává se, že dispečer zpracuje většinu zpráv, které zjistí, použijte volby MQGMO_BROTE_FIRST, MQGMO_UNMARKED_BERWSE_MSG a MQGMO_MARK_BROWSE_CO_OP (nebo MQGMO_BROWSE_CO_OP). Pokud dispečer prohlíží zprávu, kterou nemůže zpracovat, zruší označení této zprávy voláním příkazu MQGET s volbou MQMO_MATCH_MSG_TOKEN, MQGMO_UNMARK_BROWSE_CO_OP a MsgToken vráceným dříve.

Některé případy, kdy se volání MQGET nezdaří

Změní-li se některé atributy fronty použitím volby FORCE u příkazu mezi zadáním volání MQOPEN a MQGET, volání MQGET selže a vrátí kód příčiny MQRC_OBJECT_CHANGED.

Správce front označí obslužnou rutinu objektu jako neplatnou. K tomu dojde také v případě, že se změny použijí na každou frontu, na kterou je název fronty vyřešen. Atributy, které ovlivňují popisovač tímto způsobem, jsou uvedeny v popisu volání MQOPEN v MQOPEN. Pokud vaše volání vrátí kód příčiny MQRC_OBJECT_CHANGED, zavřete frontu a znovu ji otevřete a poté se pokuste zprávu znovu získat.

Pokud operace get jsou blokovány pro frontu, ze které se pokoušíte získat zprávy (nebo frontu, na kterou má být název fronty rozpoznán), volání MQGET selže a vrátí kód příčiny MQRC_GET_INHIBITED. K tomu dojde dokonce i v případě, že používáte volání MQGET pro procházení. Možná budete moci úspěšně získat zprávu, pokud se pokusíte o volání MQGET později, je-li návrh aplikace takový, že jiné programy pravidelně mění atributy front.

Byla-li dynamická fronta (dočasná nebo trvalá) odstraněna, volání MQGET s použitím dříve získaného manipulátoru objektu selžou a vrátí kód příčiny MQRC_Q_DELETED.

Zapisování aplikací typu publikování/odběr

Začněte psát aplikace pro publikování/odběr aplikací produktu IBM MQ .

Přehled koncepcí publikování/odběru naleznete v tématu Systém zpráv publikování/odběru.

Informace o psaní různých typů aplikací pro publikování/odběr naleznete v následujících tématech:

- “Zapisování aplikací vydavatele” na stránce 775
- “Zapisování aplikací odběratele” na stránce 782
- “Životní cykly publikování/odběru” na stránce 798
- “Vlastnosti zprávy publikování/odběru” na stránce 803
- “Uspořádání zpráv” na stránce 804
- “Zachycení publikací” na stránce 805
- “Volby publikování” na stránce 813
- “Volby odběru” na stránce 813

Související pojmy

“Koncepty vývoje aplikací” na stránce 6
K zápisu aplikací IBM MQ můžete použít volbu procedurálních nebo objektově orientovaných jazyků. Odkazy v tomto tématu použijte pro informace o koncepcích produktu IBM MQ , které jsou užitečné pro vývojáře aplikací.

“Vyvíjení aplikací pro IBM MQ” na stránce 5
Můžete vyvíjet aplikace k odesílání a přijímání zpráv a ke správě správců front a souvisejících prostředků. IBM MQ podporuje aplikace napsané v mnoha různých jazycích a rámcích.

“Aspekty návrhu pro aplikace produktu IBM MQ” na stránce 43
Když se rozhodnete, jak vaše aplikace mohou využívat výhody platforem a prostředí, které máte k dispozici, musíte se rozhodnout, jak používat funkce nabízené produktem IBM MQ.

“Psaní procedurální aplikace pro řazení do fronty” na stránce 689
Prostřednictvím těchto informací můžete získat informace o vytváření aplikací pro řazení do front, připojování a odpojování od správce front, publikování a odběru a otevírání a zavírání objektů.

“Zápis procedurálních aplikací klienta” na stránce 874
Co potřebujete vědět, chcete-li psát klientské aplikace na serveru IBM MQ pomocí procedurálního jazyka.

“Sestavení procedurální aplikace” na stránce 966
Aplikaci IBM MQ můžete psát v jednom z několika procedurálních jazyků a aplikaci spustit na několika různých platformách.

“Obsluha chyb procedurálních programů” na stránce 1015
Tyto informace vysvětlují chyby související s voláními MQI MQI buď při volání, nebo při doručení její zprávy do konečného cíle.

Související úlohy

“Použití ukázkových procedurálních programů produktu IBM MQ” na stránce 1034
Tyto ukázkové programy jsou napsány v procedurálních jazycích a demonstrují typická použití rozhraní MQI (Message Queue Interface). Programy produktu IBM MQ na různých platformách.

“Vývoj webových služeb pomocí produktu IBM MQ” na stránce 1255
Můžete vyvíjet aplikace produktu IBM MQ pro webové služby pomocí přenosu IBM MQ pro protokol SOAP.

Zapisování aplikací vydavatele

Začněte pracovat se zápisem aplikací vydavatele tak, že prostudujete dva příklady. První je modelován co nejpřesněji v okamžiku, kdy aplikace boduje aplikaci do fronty, a druhý demonstruje vytváření témat dynamicky-běžnější vzor pro aplikace vydavatele.

Zápis jednoduché aplikace vydavatele IBM MQ je stejně jako zápis bodu IBM MQ do bodu aplikace, který umísťuje zprávy do fronty (Tabulka 107 na stránce 776). Rozdíl spočívá v tom, že se zprávy MQPUT změní na téma, ne do fronty.

Tabulka 107. Vzor bodu k bodu versus publikování/odběr programu IBM MQ .		
Krok	**Bod připojení k bodu MQ**	**Publikovat volání MQ**
Připojit se ke správci front	MQCONN	MQCONN
Otevřít frontu	MQOPEN	
Otevřít téma		MQOPEN
Vložit zprávu (y)	MQPUT	MQPUT
Zavřít téma		MQCLOSE
Uzavřít frontu	MQCLOSE	
Odpojit od správce front	MQDISC	MQDISC

K tomu, aby se tento beton, existují dva příklady žádostí o zveřejnění cen akcií. V prvním příkladu (“Příklad 1: Vydavatel na pevné téma” na stránce 776), který je podrobně modelován při vkládání zpráv do fronty, vytvoří administrátor definici tématu podobným způsobem, jak vytvořit frontu. Programátorské kódy MQPUT slouží k zápisu zpráv na téma místo jejich zápisu do fronty. Ve druhém příkladě (“Příklad 2: Vydavatel na téma proměnné” na stránce 779) je vzorec interakce programu s IBM MQ podobný. Rozdíl je v tom, že programátor poskytuje téma, do něhož je zpráva zapisována, spíše než administrátor. V praxi to obvykle znamená, že řetězec tématu je definován jako obsah nebo je poskytován jiným zdrojem, jako např. lidským vstupem prostřednictvím prohlížeče.

Související pojmy

“Zapisování aplikací odběratele” na stránce 782
Začínáme se zápisem aplikací odběratele zkoumáním tří příkladů: Aplikace produktu IBM MQ spotřebovávající zprávy z fronty, aplikace, která vytváří odběr a nevyžaduje žádné znalosti řazení do front, a konečně příklad, který používá řazení do front i odběry.

Související informace

DEFINOVAT TÉMA
ZOBRAZIT TÉMA
ZOBRAZIT STAV TPSTATUS

Příklad 1: Vydavatel na pevné téma
Program IBM MQ pro ilustraci publikování na administrativně definované téma.

Poznámka: Styl kompaktního kódování je určen pro přehlednost, nikoli pro použití v produkčním prostředí.

Viz výstup v části Obrázek 75 na stránce 777

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <cmqc.h>
int main(int argc, char **argv)
{
    char     topicNameDefault[]   = "IBMSTOCKPRICE";
    char     publicationDefault[] = "129";
    MQCHAR48 qmName = "";

    MQHCONN  Hconn = MQHC_UNUSABLE_HCONN; /* connection handle                    */
    MQHOBJ   Hobj = MQHO_NONE;            /* object handle sub queue              */
    MQLONG   CompCode = MQCC_OK;          /* completion code                      */
    MQLONG   Reason = MQRC_NONE;          /* reason code                          */
    MQOD     td = {MQOD_DEFAULT};         /* Object descriptor                    */
    MQMD     md = {MQMD_DEFAULT};         /* Message Descriptor                   */
    MQPMO   pmo = {MQPMO_DEFAULT};        /* put message options                  */
    MQCHAR   resTopicStr[151];            /* Returned vale of topic string        */
    char *   topicName   = topicNameDefault;
    char *   publication = publicationDefault;
    memset   (resTopicStr, 0 , sizeof(resTopicStr));

    switch(argc){                         /* replace defaults with args if provided */
        default:
            publication = argv[2];
        case(2):
            topicName = argv[1];
        case(1):
            printf("Optional parameters:  TopicObject Publication\n");
    }
    do {
        MQCONN(qmName, &Hconn, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        td.ObjectType = MQOT_TOPIC;        /* Object is a topic                    */
        td.Version = MQOD_VERSION_4;       /* Descriptor needs to be V4            */
        strncpy(td.ObjectName, topicName, MQ_TOPIC_NAME_LENGTH);
        td.ResObjectString.VSPtr = resTopicStr;
        td.ResObjectString.VSBufSize = sizeof(resTopicStr)-1;
        MQOPEN(Hconn, &td, MQOO_OUTPUT | MQOO_FAIL_IF_QUIESCING, &Hobj, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        pmo.Options = MQPMO_FAIL_IF_QUIESCING | MQPMO_RETAIN;
        MQPUT(Hconn, Hobj, &md, &pmo, (MQLONG)strlen(publication)+1, publication, &CompCode,
&Reason);
        if (CompCode != MQCC_OK) break;
        MQCLOSE(Hconn, &Hobj, MQCO_NONE, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        MQDISC(&Hconn, &CompCode, &Reason);
    } while (0);
    if (CompCode == MQCC_OK)
        printf("Published \"%s\" using topic \"%s\" to topic string \"%s\"\n",
               publication, td.ObjectName, resTopicStr);
    printf("Completion code %d and Return code %d\n", CompCode, Reason);
}
```

Obrázek 74. Jednoduchý vydavatel IBM MQ s pevným tématem.

```
X:\Publish1\Debug>PublishStock
Optional parameters:  TopicObject Publication
Published "129" using topic "IBMSTOCKPRICE" to topic string "NYSE/IBM/PRICE"
Completion code 0 and Return code 0

X:\Publish1\Debug>PublishStock IBMSTOCKPRICE 155
Optional parameters:  TopicObject Publication
Published "155" using topic "IBMSTOCKPRICE" to topic string "NYSE/IBM/PRICE"
Completion code 0 and Return code 0
```

Obrázek 75. Ukázkový výstup z příkladu prvního vydavatele

Následující vybrané řádky kódu popisují aspekty, jak napsat aplikaci vydavatele pro produkt IBM MQ.

char topicNameDefault[] = "IBMSTOCKPRICE";
V programu je definován výchozí název tématu. Můžete ji přepsat zadáním názvu jiného objektu tématu jako prvního argumentu pro program.

MQCHAR resTopicStr[151];
`resTopicStr` je uveden v `td.ResObjectString.VSPtr` a je používán produktem `MQOPEN` k vrácení vyřešeného řetězce tématu. Učinit délku `resTopicStr` o jednu větší než délku předanou v produktu `td.ResObjectString.VSBufSize` za účelem poskytnutí prostoru pro ukončení s hodnotou null.

memset (resTopicStr, 0, sizeof(resTopicStr));
Inicializujte `resTopicStr` na hodnotu null, abyste zajistili, že přeložený řetězec tématu vrácený v `MQCHARV` má hodnotu null ukončen.

td.ObjectType = MQOT_TOPIC
Pro publikování/odběr je k dispozici nový typ objektu: *objekt tématu*.

td.Version = MQOD_VERSION_4;
Chcete-li použít nový typ objektu, musíte použít alespoň *verzi 4* deskriptoru objektu.

strncpy(td.ObjectName, topicName, MQ_OBJECT_NAME_LENGTH);
`topicName` je název objektu tématu, který se někdy označuje jako objekt administrativního tématu. V tomto příkladu je třeba objekt tématu vytvořit předem pomocí Průzkumníka IBM MQ nebo tohoto příkazu MQSC,

```
DEFINE TOPIC(IBMSTOCKPRICE) TOPICSTR(NYSE/IBM/PRICE) REPLACE;
```

td.ResObjectString.VSPtr = resTopicStr;
Vyřešený řetězec tématu se vypisuje v konečném `printf` v programu. Nastavte strukturu `MQCHARV` `ResObjectString` pro IBM MQ , aby vracel vyřešený řetězec zpět do programu.

MQOPEN(Hconn, &td, MQOO_OUTPUT | MQOO_FAIL_IF_QUIESCING, &Hobj, &CompCode, &Reason);
Otevřít téma pro výstup; stejně jako otevření fronty pro výstup.

pmo.Options = MQPMO_FAIL_IF_QUIESCING | MQPMO_RETAIN;
Chcete, aby byli noví odběratelé schopni přijímat publikování, a uvedením `MQPMO_RETAIN` v vydavateli, když spustíte odběratele, obdrží nejnovější publikování, publikovaná před spuštěním odběratele, jako první odpovídající publikování. Alternativou je poskytnout odběratelům publikování publikovaná pouze poté, co je odběratel spuštěn. Kromě toho má odběratel možnost odmítnout přijetí zachovaného publikování zadáním `MQSO_NEW_PUBLICATIONS_ONLY` ve svém odběru.

MQPUT(Hconn, Hobj, &md, &pmo, (MQLONG)strlen(publication)+1, publication, &CompCode, &Reason);
Přidejte 1 až do délky řetězce předaného do `MQPUT` , čímž se předá znak ukončení null do IBM MQ jako součást vyrovnávací paměti zpráv.

Co první příklad demonstruje? Příklad imituje co nejvíce zkuseného a testovaného tradičního vzoru pro zápis bodu IBM MQ programů. Důležitým rysem programovacího modelu produktu IBM MQ je to, že programátor se nezabývá tím, kam se zprávy odesílají. Úkolem programátora je připojit se ke správci front a předat mu zprávy, které mají být distribuovány příjemcům. V paradigmatu mezi dvěma body programátor otevře frontu (pravděpodobně alias frontu alias), kterou administrátor nakonfiguroval. Fronta aliasů směruje zprávy do cílové fronty, a to buď na lokálním správci front, nebo na vzdáleném správci front. Zatímco zprávy čekají na doručení, jsou uloženy ve frontách někde mezi zdrojem a místem určení.

Místo otevření fronty otevře programátor v rámci vzoru publikování/odběru téma. V našem příkladu je téma asociováno s řetězcem tématu administrátorem. Správce front předá publikování s použitím front lokálním nebo vzdáleným odběratelům, kteří mají odběry odpovídající řetězci tématu publikování. Pokud jsou publikace uchovávány, uchovává správce front nejnovější kopii této publikace, a to i v případě, že dosud nemá žádné odběratele. Zachované publikování je k dispozici pro postoupení budoucím odběratelům. Aplikace vydavatele nehraje žádnou roli při výběru nebo směrování publikování do místa určení; jejím úkolem je vytvářet a vkládat publikace do témat definovaných administrátorem.

Tento příklad fixního tématu je atypický pro mnoho aplikací typu publikování/odběr: je statický. Vyžaduje to, aby administrátor definoval řetězce témat a změnil témata, která jsou publikována. Aplikace publish/subscribe publish-subscribe musí znát některé nebo všechny stromy témat. Možná se témata často mění, nebo snad i když se témata příliš nemění, počet kombinací témat je velký a pro administrátora je příliš obtížné definovat uzel tématu pro každý řetězec tématu, který může být potřeba publikovat. Možná řetězce tématu nejsou známy v předstihu publikování; aplikace vydavatele může použít informace z obsahu publikování k určení řetězce tématu, nebo může obsahovat informace o řetězcích témat pro publikování z jiného zdroje, jako je například vstup z lidských zdrojů z prohlížeče. Následující příklad ukazuje, jak dynamicky vytvářet témata jako součást aplikace vydavatele k zajišťování dynamičtějších stylů publikování.

Témata, vydavatelé a odběratelé dohromady. Navrhování pravidel nebo architektury pro pojmenovávání témat a jejich uspořádání do stromů témat je důležitým krokem při vývoji řešení typu publikování-odběr. Pečlivě se podívejte na to, do jaké míry se organizace stromu témat spojí s vydavatelem a programy odběratele dohromady a spojí je s obsahem stromu témat. Zeptejte se sami sebe na otázku, zda změny ve stromu témat ovlivňují vydavatele a aplikace odběratele, a jak můžete minimalizovat efekt. Zabudovaná do architektury modelu publikování/odběru IBM MQ je představou objektu administrativního tématu, který poskytuje základní část nebo kořenový podstrom tématu. Objekt tématu vám dává možnost definovat kořenovou část stromu témat administrativně, která zjednodušuje programování a provoz aplikací, a následně vylepšuje udržovatelnost. Pokud například implementujete více aplikací publikování/odběru, které mají izolované stromy témat, pak administrativně definováním kořenové části stromu témat, můžete zaručit izolaci stromu témat i v případě, že neexistuje konzistence v konvencích pojmenování tématu, které byly přijaty různými aplikacemi.

V praxi se aplikace vydavatele týkají spektra výhradně pomocí pevných témat, podobně jako v tomto příkladu, a proměnných témat, jako v následujícím příkladu. Produkt “Příklad 2: Vydavatel na téma proměnné” na stránce 779 také demonstruje kombinaci použití témat a řetězců témat.

Související pojmy

“Příklad 2: Vydavatel na téma proměnné” na stránce 779
Program WebSphere MQ pro ilustraci publikování v rámci programově definovaného tématu.

“Zapisování aplikací odběratele” na stránce 782
Začínáme se zápisem aplikací odběratele zkoumáním tří příkladů: Aplikace produktu IBM MQ spotřebovávající zprávy z fronty, aplikace, která vytváří odběr a nevyžaduje žádné znalosti řazení do front, a konečně příklad, který používá řazení do front i odběry.

Příklad 2: Vydavatel na téma proměnné
Program WebSphere MQ pro ilustraci publikování v rámci programově definovaného tématu.

Poznámka: Styl kompaktního kódování je určen pro přehlednost, nikoli pro použití v produkčním prostředí.

Viz výstup v části Obrázek 77 na stránce 780.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <cmqc.h>
int main(int argc, char **argv)
{
    char     topicNameDefault[]   = "STOCKS";
    char     topicStringDefault[] = "IBM/PRICE";
    char     publicationDefault[] = "130";
    MQCHAR48 qmName = "";

    MQHCONN  Hconn = MQHC_UNUSABLE_HCONN; /* connection handle           */
    MQHOBJ   Hobj     = MQHO_NONE;   /* object handle sub queue          */
    MQLONG   CompCode = MQCC_OK;     /* completion code                  */
    MQLONG   Reason   = MQRC_NONE;   /* reason code                      */
    MQOD     td = {MQOD_DEFAULT};    /* Object descriptor                */
    MQMD     md = {MQMD_DEFAULT};    /* Message Descriptor               */
    MQPMO   pmo = {MQPMO_DEFAULT};   /* put message options              */
    MQCHAR   resTopicStr[151];       /* Returned value of topic string */
    char *   topicName   = topicNameDefault;
    char *   topicString = topicStringDefault;
    char *   publication = publicationDefault;
    memset   (resTopicStr, 0 , sizeof(resTopicStr));

    switch(argc){            /* Replace defaults with args if provided */
        default:
            publication = argv[3];
        case(3):
            topicString = argv[2];
        case(2):
            if (strcmp(argv[1],"/"))  /* "/" invalid = No topic object  */
                topicName = argv[1];
            else
                *topicName = '\0';
        case(1):
            printf("Provide parameters:  TopicObject TopicString Publication\n");
    }

    printf("Publish \"%s\" to topic \"%-.48s\" and topic string \"%s\"\n", publication, topicName,
topicString);
    do {
        MQCONN(qmName, &Hconn, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        td.ObjectType = MQOT_TOPIC;    /* Object is a topic             */
        td.Version = MQOD_VERSION_4;   /* Descriptor needs to be V4     */
        strncpy(td.ObjectName, topicName,  MQ_TOPIC_NAME_LENGTH);
        td.ObjectString.VSPtr = topicString;
        td.ObjectString.VSLength = (MQLONG)strlen(topicString);
        td.ResObjectString.VSPtr = resTopicStr;
        td.ResObjectString.VSBufSize = sizeof(resTopicStr)-1;
        MQOPEN(Hconn, &td, MQOO_OUTPUT | MQOO_FAIL_IF_QUIESCING, &Hobj, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        pmo.Options = MQPMO_FAIL_IF_QUIESCING | MQPMO_RETAIN;
        MQPUT(Hconn, Hobj, &md, &pmo, (MQLONG)strlen(publication)+1, publication, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        MQCLOSE(Hconn, &Hobj, MQCO_NONE, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        MQDISC(&Hconn, &CompCode, &Reason);
    } while (0);
    if (CompCode == MQCC_OK)
        printf("Published \"%s\" to topic string \"%s\"\n",  publication, resTopicStr);
    printf("Completion code %d and Return code %d\n", CompCode, Reason);
}
```

Obrázek 76. Jednoduchý vydavatel IBM MQ na téma proměnné.

```
X:\Publish2\Debug>PublishStock
Provide parameters: TopicObject TopicString Publication
Publish "130" to topic "STOCKS" and topic string "IBM/PRICE"
Published "130" to topic string "NYSE/IBM/PRICE"
Completion code 0 and Return code 0

X:\Publish2\Debug>PublishStock / NYSE/IBM/PRICE 131
Provide parameters: TopicObject TopicString Publication
Publish "131" to topic "" and topic string "NYSE/IBM/PRICE"
Published "131" to topic string "NYSE/IBM/PRICE"
Completion code 0 and Return code 0
```

Obrázek 77. Příklad výstupu ukázky z druhého vydavatele

Je zde několik poznámek k tomuto příkladu.

char topicNameDefault[] = "STOCKS";
Výchozí název tématu STOCKS definuje část řetězce tématu. Tento název tématu můžete přepsat tak, že jej poskytnete jako první argument programu, nebo jej odstraňte zadáním parametru / jako prvního parametru.

char topicString[101] = "IBM/PRICE";
IBM/PRICE je výchozí řetězec tématu. Tento řetězec tématu můžete přepsat tím, že jej poskytnete jako druhý argument programu.

Správce front kombinuje řetězec tématu poskytovaný objektem tématu STOCKS , "NYSE", s řetězcem tématu poskytnutým programem "IBM/PRICE" a vkládá mezi tyto dva řetězce témat "/" . Výsledkem je vyřešený řetězec tématu "NYSE/IBM/PRICE". Výsledný řetězec tématu je stejný jako řetězec definovaný v objektu tématu produktu IBMSTOCKPRICE a má přesně stejný účinek.

Objekt administrativního tématu přidružený k vyřešenému řetězci tématu nemusí být nutně totožný s objektem tématu, který byl předán vydavateli MQOPEN . Produkt IBM MQ používá strom implicitně v analyzovaného řetězci tématu k práci, který objekt administračních témat definuje atributy přidružené k této publikaci.

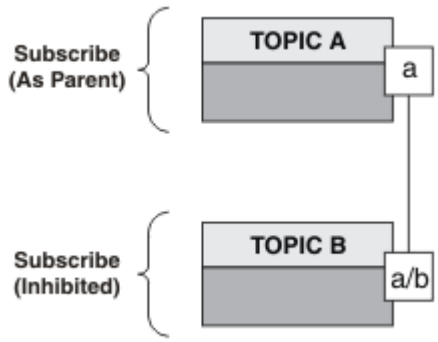
Předpokládejme, že existují dva objekty témat A a B, a A definuje téma "a"a B definuje téma "a/b" (Obrázek 78 na stránce 781). Pokud se program vydavatele odkazuje na objekt tématu A a poskytuje řetězec tématu "b", interpretujete téma na řetězec tématu "a/b", pak publikování dědí vlastnosti z objektu tématu B , protože téma se shoduje s řetězcem tématu "a/b" definovaným pro B.

if (strcmp(argv[1],"/"))
argv[1] je volitelně zadán parametr topicName. "/" je neplatný jako název tématu; zde označuje, že neexistuje žádné jméno tématu, a řetězec tématu je poskytován zcela programem. Výstup v produktu Obrázek 77 na stránce 780 zobrazuje řetězec celého tématu dynamicky dodávaný programem.

strncpy(td.ObjectName, topicName, MQ_OBJECT_NAME_LENGTH);
V případě výchozího případu je třeba, aby byl volitelný topicName vytvořen předem pomocí Průzkumníka IBM MQ nebo tohoto příkazu MQSC:

```
DEFINE TOPIC(STOCKS) TOPICSTR(NYSE) REPLACE;
```

td.ObjectString.VSPtr = topicString;
Řetězec tématu je pole MQCHARV v deskriptoru tématu



Obrázek 78. Přidružení objektů témat

Co druhý příklad demonstruje? Ačkoli je kód velmi podobný prvnímu příkladu-efektivně existují pouze dva řádky-výsledek je výrazně odlišný od prvního programu. Programátor určuje místa určení, do kterých jsou odesílána publikace. Ve spojení s minimálním vstupem administrátora použitým k návrhu aplikací odběratele, není třeba předem definovat žádná témata nebo fronty pro směrování publikací od vydavatelů k odběratelům.

V paradigmatu systému zpráv typu point-to-point je třeba definovat fronty před tím, než je možné zprávy směrovat. U publish/subscribe to nečiní, ačkoli produkt IBM MQ implementuje publikování/odběr pomocí systému řazení do front základního systému; výhody zaručeného doručení, transakcese a volné vazby přidružené k systému zpráv a řazení do fronty jsou zděděny aplikacemi publikování/odběru.

Vývojář musí rozhodnout, zda vydavatel a odběratel mají být informováni o základním stromu témat či nikoli, a také zda si programy odběratelů vědí o řazení do fronty či nikoli. Studium ukázkových aplikací odběratele další. Jsou navrženy tak, aby byly používány s příklady vydavatele, obvykle publikováním a přihlášením k odběru `NYSE/IBM/PRICE`.

Související pojmy

“Příklad 1: Vydavatel na pevné téma” na stránce 776
Program IBM MQ pro ilustraci publikování na administrativně definované téma.

“Zapisování aplikací odběratele” na stránce 782
Začínáme se zápisem aplikací odběratele zkoumáním tří příkladů: Aplikace produktu IBM MQ spotřebovávající zprávy z fronty, aplikace, která vytváří odběr a nevyžaduje žádné znalosti řazení do front, a konečně příklad, který používá řazení do front i odběry.

Zapisování aplikací odběratele

Začínáme se zápisem aplikací odběratele zkoumáním tří příkladů: Aplikace produktu IBM MQ spotřebovávající zprávy z fronty, aplikace, která vytváří odběr a nevyžaduje žádné znalosti řazení do front, a konečně příklad, který používá řazení do front i odběry.

V produktu Tabulka 108 na stránce 782 jsou uvedeny tři styly odběratele nebo odběratele, společně s posloupnostmi volání funkcí IBM MQ , které je charakterizují.

1. První styl MQ Publication Consumer je identický s bodem k bodu MQ , který provádí pouze MQGET. Aplikace nemá žádné informace o tom, že se jedná o využití publikací-stačí číst zprávy z fronty. Odběr, který způsobí přesměrování publikování do fronty, je vytvořen administrativně pomocí Průzkumníka IBM MQ nebo příkazu.
2. Druhý styl je upřednostňovaným vzorem pro většinu aplikací odběratele. Aplikace odběratele vytvoří odběr a poté získá publikace. Správa front je prováděna správcem front.
3. Ve třetím stylu se aplikace odběratele rozhodne otevřít a zavřít základní frontu, která se používá pro publikování, a také vydávat odběry pro vyplnění fronty s publikacemi.

Jedním ze způsobů, jak porozumět těmto stylům, je prostudovat si vzorové programy C uvedené v Tabulka 108 na stránce 782 pro každý ze stylů. Příklady jsou navrženy ke spuštění ve spojení s příkladem vydavatele nalezeným v souboru “Zapisování aplikací vydavatele” na stránce 775.

Tabulka 108. Vzory programu IBM MQ pro bod k odběru a odběr.

Krok	**Spotřebitel zpráv MQ**	**“Příklad 1: Spotřebitel publikování MQ” na stránce 783**	**“Příklad 2: Spravovaný odběratel MQ” na stránce 785**	**“Příklad 3: Nespravovaný odběratel MQ” na stránce 790**
Připojit se ke správci front	MQCONN	MQCONN	MQCONN	MQCONN
Otevřít frontu	MQOPEN	MQOPEN		MQOPEN
Odebírat			MQSUB	MQSUB
Získat zprávu (y)	MQGET	MQGET	MQGET	MQGET
Uzavřít frontu	MQCLOSE	MQCLOSE	(MQCLOSE)	MQCLOSE
Zavřít odběr			MQCLOSE	MQCLOSE
Odpojit od správce front	MQDISC	MQDISC	MQDISC	MQDISC

Použití funkce MQCLOSE je vždy volitelné, buď pro uvolnění prostředků, předání voleb MQCLOSE nebo pouze pro symetrii s MQOPEN. Vzhledem k tomu, že není pravděpodobné, že je třeba určit volby MQCLOSE, je-li fronta odběru uzavřena v případu odběratele spravovaných MQ a argument symetrie není relevantní, fronta odběru není v produktu Příklad 2: Spravovaný odběratel produktu MQexplicitně uzavřena.

Jiný způsob, jak porozumět vzorům aplikací publikování/odběru, je také příliš pohledem na interakce mezi různými zúčastněnými objekty. Linie, nebo sekvenční diagramy UML jsou dobrým způsobem ke studiu interakcí. Tři příklady linie linie jsou popsány v “Životní cykly publikování/odběru” na stránce 798.

Příklad 1: Spotřebitel publikování MQ
Spotřebitel zpráv produktu MQ je spotřebitel zpráv produktu IBM MQ , který se nepřihlašují k odběru vlastních témat.

Chcete-li vytvořit odběr a frontu publikování pro tento příklad, spusťte následující příkazy nebo definujte objekty pomocí Průzkumníka IBM MQ .

```
DEFINE QLOCAL(STOCKTICKER) REPLACE;
DEFINE SUB(IBMSTOCKPRICESUB) DEST(STOCKTICKER) TOPICOBJ(IBMSTOCKPRICE) REPLACE;
```

Odběr produktu IBMSTOCKPRICESUB se odkazuje na objekt tématu produktu IBMSTOCK vytvořený pro příklad vydavatele a na lokální frontu STOCKTICKER. Objekt tématu IBMSTOCK definuje řetězec tématu, který se používá v odběru NYSE/IBM/PRICE. Všimněte si, že objekt tématu a fronta použitá pro příjem publikování musí být definována před vytvořením odběru.

Šablona spotřebitele publikování MQ obsahuje mnoho cenných fazet:

1. Vícenásobné zpracování: sdílení mimo práci na čtení publikací. Všechny publikace se nacházejí v jedné frontě přidružené k odběru tématu odběru. Frontu s použitím produktu MQOO_INPUT_SHAREDmůže otevřít více spotřebitelů.
2. Centrálně spravované odběry. Aplikace nesestavují svá vlastní témata odběru nebo odběry; administrátor je odpovědný za místa odeslání publikací.
3. Spojení odběru: více různých odběrů lze odeslat do jediné fronty.
4. Trvalost odběru: Fronta přijímá všechna publikování bez ohledu na to, zda jsou či nejsou aktivní spotřebitelé.
5. Migrace a koexistence: kód odběratele funguje stejně dobře pro dvoubodový systém a scénář publikování/odběru.

Odběr vytvoří vztah mezi řetězcem tématu NYSE/IBM/PRICE a frontou STOCKTICKER. Publikace, včetně všech aktuálně zachovaných publikací, jsou od okamžiku vytvoření odběru předány produktu STOCKTICKER .

Administrativně vytvořený odběr může být spravován nebo nespravovaný. Spravovaný odběr se projeví, jakmile byl vytvořen, stejně jako nespravovaný odběr. Ne všechny fazety vzorku jsou k dispozici pro spravovaný odběr. Viz “Příklad 3: Nespravovaný odběratel MQ” na stránce 790

Poznámka: Styl kompaktního kódování je určen pro přehlednost, nikoli pro použití v produkčním prostředí.

Výsledky jsou zobrazeny v Obrázek 80 na stránce 784.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <cmqc.h>
int main(int argc, char **argv)
{
MQCHAR     publicationBuffer[101];
MQCHAR48 subscriptionQueueDefault = "STOCKTICKER";
MQCHAR48 qmName = "";            /* Use default queue manager  */

MQHCONN Hconn = MQHC_UNUSABLE_HCONN;    /* connection handle      */
MQHOBJ  Hobj = MQHO_NONE;         /* object handle sub queue   */
MQLONG  CompCode = MQCC_OK;        /* completion code       */
MQLONG  Reason = MQRC_NONE;        /* reason code         */
MQLONG  messlen = 0;
MQOD   od = {MQOD_DEFAULT};        /* Unmanaged subscription queue */
MQMD   md = {MQMD_DEFAULT};        /* Message Descriptor      */
MQGMO  gmo = {MQGMO_DEFAULT};       /* Get message options     */
char *  publication=publicationBuffer;
char *  subscriptionQueue = subscriptionQueueDefault;

switch(argc){           /* Replace defaults with args if provided */
default:
subscriptionQueue = argv[1]
case(1):
printf("Optional parameter: subscriptionQueue\n");
}

do {
MQCONN(qmName, &Hconn, &CompCode, &Reason);
if (CompCode != MQCC_OK) break;
strncpy(od.ObjectName, subscriptionQueue, MQ_Q_NAME_LENGTH);
MQOPEN(Hconn, &od, MQOO_INPUT_AS_Q_DEF | MQOO_FAIL_IF_QUIESCING , &Hobj, &CompCode, &Reason);
if (CompCode != MQCC_OK) break;
gmo.Options = MQGMO_WAIT | MQGMO_NO_SYNCPOINT | MQGMO_CONVERT;
gmo.WaitInterval = 10000;
printf("Waiting %d seconds for publications from %s\n", gmo.WaitInterval/1000,
subscriptionQueue);
do {
memcpy(md.MsgId, MQMI_NONE, sizeof(md.MsgId));
memcpy(md.CorrelId, MQCI_NONE, sizeof(md.CorrelId));
md.Encoding    = MQENC_NATIVE;
md.CodedCharSetId = MQCCSI_Q_MGR;
memset(publication, 0, sizeof(publicationBuffer));
MQGET(Hconn, Hobj, &md, &gmo, sizeof(publicationBuffer)-1, publication, &messlen,
&CompCode, &Reason);
if (Reason == MQRC_NONE)
printf("Received publication \"%s\"\n", publication);
}
while (CompCode == MQCC_OK);
if (CompCode != MQCC_OK && Reason != MQRC_NO_MSG_AVAILABLE) break;
MQCLOSE(Hconn, &Hobj, MQCO_NONE, &CompCode, &Reason);
if (CompCode != MQCC_OK) break;
MQDISC(&Hconn, &CompCode, &Reason);
} while (0);
printf("Completion code %d and Return code %d\n", CompCode, Reason);
}
```

Obrázek 79. Spotřebitel publikování MQ .

```
X:\Subscribe1\Debug>Subscribe1
Optional parameter: subscriptionQueue
Waiting 10 seconds for publications from STOCKTICKER
Received publication "129"
Completion code 0 and Return code 0
```

Obrázek 80. Výstup ze spotřebitele publikování MQ

Existuje několik standardních programovacích tipů pro jazyk IBM MQ *C* , které mají být informovány o:

memset(publication, 0, sizeof(publicationBuffer));
Ujistěte se, že zpráva má koncovou hodnotu null pro snadné formátování pomocí `printf`. Příklad vydavatele zahrnuje koncovou hodnotu null ve vyrovnávací paměti zpráv předané produktu MQPUT přidáním 1 do `strlen(publication)`. Nastavení vyrovnávacích pamětí MQCHAR na hodnotu null je dobrým programovacím stylem pro programy IBM MQ C, které používají vyrovnávací paměti k ukládání řetězců a zajišťují, že hodnota null bude obsahovat pole znaků, které nevyplní vyrovnávací paměť zcela.

MQGET(Hconn, Hobj, &md, &gmo, sizeof(publicationBuffer)-1, publication, &messlen, &CompCode, &Reason);
Vyhrazení jedné hodnoty null na konci vyrovnávací paměti zpráv zajistí, že vrácená zpráva má koncovou hodnotu null v případě, že `if (messlen == strlen(publication));` je true. Tento tip doplňuje předchozí a zajišťuje, že v `publicationBuffer` existuje alespoň jedna hodnota null, která není přepsána obsahem produktu `publication`.

Související pojmy

“Příklad 2: Spravovaný odběratel MQ” na stránce 785
Spravovaný odběratel produktu MQ je upřednostňovaným vzorem pro většinu aplikací odběratele. Příklad vyžaduje *ne* administrative definition of queues, topics or subscriptions.

“Příklad 3: Nespravovaný odběratel MQ” na stránce 790
Nespravovaný odběratel je důležitou třídou aplikace odběratele. S ní kombinujete výhody publikování/odběru s *ovládacím prvkem* řazení do fronty a spotřebou publikací. Tento příklad ukazuje různé způsoby kombinování odběrů a front.

“Zapisování aplikací vydavatele” na stránce 775
Začněte pracovat se zápisem aplikací vydavatele tak, že prostudujete dva příklady. První je modelován co nejpřesněji v okamžiku, kdy aplikace boduje aplikaci do fronty, a druhý demonstruje vytváření témat dynamicky-běžnější vzor pro aplikace vydavatele.

Příklad 2: Spravovaný odběratel MQ
Spravovaný odběratel produktu MQ je upřednostňovaným vzorem pro většinu aplikací odběratele. Příklad vyžaduje *ne* administrative definition of queues, topics or subscriptions.

Tento nejjednodušší typ spravovaného odběratele obvykle používá *netrvalý* odběr. Tento příklad se zaměřuje na netrvalý odběr. Odběr trvá pouze po dobu životnosti obslužné rutiny odběru z produktu MQSUB. Všechny publikace, které odpovídají řetězci tématu během doby životnosti odběru, jsou odeslány do fronty odběru (a případně zachované publikace, pokud příznak `MQSO_NEW_PUBLICATIONS_ONLY` není nastaven nebo je nastaven na výchozí hodnotu, byla zachována dřívější publikování odpovídající řetězci tématu a publikování byla trvalá nebo správce front nebyl ukončen, od doby, kdy byla publikace vytvořena).

S tímto vzorem můžete také použít *trvalý* odběr. Obvykle je-li použit spravovaný trvalý odběr, který je proveden z důvodů spolehlivosti, spíše než aby byl nastaven odběr, který by bez chyb při pokusu o trvalý odběr mohl přežít odběratele. Další informace o různých životních cyklech přidružených ke spravovaným, nespravovaným, trvalým a netrvalým odběrům naleznete v příslušné sekci témat.

Trvalé odběry jsou často přidruženy k trvalým publikacím a netrvalé odběry s netrvalými publikacemi, ale neexistuje žádný nutný vztah mezi trvanlivostí odběru a perzistencí publikování. Všechny čtyři kombinace perzistence a trvanlivosti jsou možné.

Pro spravovaný netrvalý případ bude správce front vytvořit frontu odběru, která je vyprázdněna a odstraněna, když je fronta zavřena. Publikování budou z fronty odebrány při zavření netrvalého odběru.

Cenné fazety spravovaného netrvanlivého vzoru, které jsou příkladem tohoto kódu, jsou následující:

1. On Demand subscription: řetězec tématu odběru je dynamický. Poskytne ji aplikace, když je spuštěna.
2. Vlastní správa fronty: Fronta odběru je sama definující a spravuje.
3. Vlastní správa životního cyklu odběru: *netrvanlivé* odběry existují pouze po dobu trvání aplikace odběratele.
   - Definujete-li *trvalý* odběr, bude jeho platnost uložena do trvalé fronty odběru a budou do ní nadále ukládány publikace bez aktivních programů odběratele. Správce front tuto frontu odstraní (a vymaže

z ní všechny nenačtené publikace) až poté, co se aplikace nebo administrátor rozhodne odstranit odběr. Odběr lze odstranit pomocí administrativního příkazu nebo uzavřením odběru s použitím volby MQCO_REMOVE_SUB .

- Zvažte nastavení SubExpiry pro trvalé odběry, aby publikování přestaly být odesílána do fronty a odběratel může spotřebovat všechny zbývající publikace před odebráním odběru a způsobit, že správce front odstraní frontu a všechny zbývající publikace na ní.

4. Flexibilní implementace řetězce témat: Správa témat odběru je zjednodušena definováním základní části odběru pomocí administrativně definovaného tématu. Kořenová část stromu témat je poté skryta z aplikace. Skrytím kořenové části lze aplikaci implementovat bez neúmyslné vytvoření stromu témat, který se překrývá s jiným stromem témat vytvořeným jinou instancí, nebo jinou aplikací.
5. Administrované témata: pomocí řetězce tématu, ve kterém se první část shoduje s administrativně definovaným objektem tématu, jsou publikace spravovány v souladu s atributy objektu tématu.
   - Pokud například první část řetězce tématu odpovídá řetězci tématu přidruženému k objektu klastrovaného tématu, může odběr přijímat publikování od jiných členů klastru.
   - Selektivní shoda administrativně definovaných objektů témat a programově definovaných odběrů umožňuje kombinovat výhody obou. Administrátor poskytuje atributy pro témata a programátor dynamicky definuje dílčí témata bez toho, že by se měl zajímat o správu témat.
   - Jedná se o výsledný řetězec tématu, který se používá ke shodě s objektem tématu, který poskytuje atributy přidružené k tématu, a ne nutně objekt tématu pojmenovaný v sd.Objectname, ačkoli se obvykle změní na jeden a stejný. Viz “Příklad 2: Vydavatel na téma proměnné” na stránce 779.

Díky tomu, že odběr bude trvalý v tomto příkladu, budou publikování nadále odesílána do fronty odběru poté, co odběratel uzavřel odběr s použitím volby MQCO_KEEP_SUB . Fronta pokračuje v přijímání publikování, pokud není aktivní odběratel. Toto chování můžete potlačit vytvořením odběru s použitím volby MQSO_PUBLICATIONS_ON_REQUEST a použitím produktu MQSUBRQ požadovat zachované publikování.

Odběr lze obnovit později otevřením odběru s použitím volby MQCO_RESUME .

Můžete použít popisovač fronty Hobj, vrácený produktem MQSUB v mnoha ohledech. Popisovač fronty se používá v příkladu k zjišťování názvu fronty odběru. Spravované fronty se otevírají s použitím výchozích modelových front SYSTEM.NDURABLE.MODEL.QUEUE nebo SYSTEM.DURABLE.MODEL.QUEUE. Výchozí nastavení můžete přepsat zadáním vlastních trvalých a netrvalých modelových front na téma podle jednotlivých témat jako vlastností objektu tématu přidruženého k odběru.

Bez ohledu na atributy zděděné z modelových front nelze znovu použít obslužnou rutinu spravované fronty k vytvoření dalšího odběru. Další manipulační prostředek pro spravovanou frontu nelze získat tak, že otevřete spravovanou frontu podruhé s použitím vráceného názvu fronty. Fronta se chová tak, jako kdyby byla otevřena pro výhradní vstup.

Nespravované fronty jsou flexibilnější než spravované fronty. Můžete například sdílet nespravované fronty, nebo definovat více odběrů v jedné frontě. Následující příklad ukazuje, jak kombinovat odběry s nespravovanou frontou odběrů.

Poznámka: Styl kompaktního kódování je určen pro přehlednost, nikoli pro použití v produkčním prostředí.

Výsledky jsou zobrazeny v Obrázek 83 na stránce 788.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <cmqc.h>

void inquireQname(MQHCONN HConn, MQHOBJ Hobj, MQCHAR48 qName);

int main(int argc, char **argv)
{
    MQCHAR48 topicNameDefault     = "STOCKS";
    char     topicStringDefault[] = "IBM/PRICE";
    MQCHAR48 qmName = "";                 /* Use default queue manager     */
    MQCHAR48 qName = "";                  /* Allocate to query queue name  */
    char     publicationBuffer[101];      /* Allocate to receive messages  */
    char     resTopicStrBuffer[151];   /* Allocate to resolve topic string */

    MQHCONN  Hconn = MQHC_UNUSABLE_HCONN; /* connection handle             */
    MQHOBJ   Hobj = MQHO_NONE;            /* publication queue handle      */
    MQHOBJ   Hsub = MQSO_NONE;            /* subscription handle           */
    MQLONG   CompCode = MQCC_OK;          /* completion code               */
    MQLONG   Reason = MQRC_NONE;          /* reason code                   */
    MQLONG   messlen = 0;
    MQSD     sd = {MQSD_DEFAULT};         /* Subscription Descriptor       */
    MQMD     md = {MQMD_DEFAULT};         /* Message Descriptor            */
    MQGMO   gmo = {MQGMO_DEFAULT};        /* get message options           */

    char *   topicName   = topicNameDefault;
    char *   topicString = topicStringDefault;
    char *   publication = publicationBuffer;
    char *   resTopicStr = resTopicStrBuffer;
    memset(resTopicStr, 0, sizeof(resTopicStrBuffer));

    switch(argc){               /* Replace defaults with args if provided */
        default:
          topicString = argv[2];
        case(2):
          if (strcmp(argv[1],"/"))       /* "/" invalid = No topic object  */
             topicName = argv[1];
          else
             *topicName = '\0';
        case(1):
             printf("Optional parameters: topicName, topicString\nValues \"%s\"  \"%s\"\n",
                   topicName, topicString);
}
```

Obrázek 81. Spravovaný odběratel MQ -část 1: deklarace a zpracování parametrů.

V tomto příkladu jsou k dispozici některé další komentáře k deklaracím.

MQHOBJ Hobj = MQHO_NONE;
Nelze explicitně otevřít netrvalou spravovanou frontu odběru, která má přijímat publikování, ale při otevření fronty pro vás správce front je třeba přidělit úložiště pro obsluhu objektů, které tento správce front obsahuje. Je důležité inicializovat popisovač na MQHO_OBJECT. Tato zpráva informuje správce front o tom, že je třeba vrátit manipulátor fronty do fronty odběru.

MQSD sd = {MQSD_DEFAULT};
Nový deskriptor odběru použitý v produktu MQSUB.

MQCHAR48 qName;
Ačkoli příklad nevyžaduje znalosti fronty odběru, příklad se dotáže na název fronty odběru-vazba MQINQ je trochu trapná v jazyku C, takže byste mohli najít část příkladu, která je užitečná ke studiu.

```
    do {
        MQCONN(qmName, &Hconn, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        strncpy(sd.ObjectName, topicName, MQ_TOPIC_NAME_LENGTH);
        sd.ObjectString.VSPtr = topicString;
        sd.ObjectString.VSLength = MQVS_NULL_TERMINATED;
        sd.Options = MQSO_CREATE | MQSO_MANAGED | MQSO_NON_DURABLE | MQSO_FAIL_IF_QUIESCING ;
        sd.ResObjectString.VSPtr = resTopicStr;
        sd.ResObjectString.VSBufSize = sizeof(resTopicStrBuffer)-1;
        MQSUB(Hconn, &sd, &Hobj, &Hsub, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        gmo.Options = MQGMO_WAIT | MQGMO_NO_SYNCPOINT | MQGMO_CONVERT;
        gmo.WaitInterval = 10000;
        inquireQname(Hconn, Hobj, qName);
        printf("Waiting %d seconds for publications matching \"%s\" from \"%-0.48s\"\n",
               gmo.WaitInterval/1000, resTopicStr, qName);
        do {
            memcpy(md.MsgId, MQMI_NONE, sizeof(md.MsgId));
            memcpy(md.CorrelId, MQCI_NONE, sizeof(md.CorrelId));
            md.Encoding       = MQENC_NATIVE;
            md.CodedCharSetId = MQCCSI_Q_MGR;
            memset(publicationBuffer, 0, sizeof(publicationBuffer));
            MQGET(Hconn, Hobj, &md, &gmo, sizeof(publicationBuffer)-1),
                publication, &messlen, &CompCode, &Reason);
            if (Reason == MQRC_NONE)
                printf("Received publication \"%s\"\n", publication);
        }
        while (CompCode == MQCC_OK);
        if (CompCode != MQCC_OK && Reason != MQRC_NO_MSG_AVAILABLE) break;
        MQCLOSE(Hconn, &Hsub, MQCO_REMOVE_SUB, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        MQDISC(&Hconn, &CompCode, &Reason);
    } while (0);
    printf("Completion code %d and Return code %d\n", CompCode, Reason);
    return;
}
void inquireQname(MQHCONN Hconn, MQHOBJ Hobj, MQCHAR48 qName) {
#define _selectors 1
#define _intAttrs 1

    MQLONG select[_selectors] = {MQCA_Q_NAME}; /* Array of attribute selectors */
    MQLONG intAttrs[_intAttrs];                /* Array of integer attributes  */
    MQLONG CompCode, Reason;
    MQINQ(Hconn, Hobj, _selectors, select, _intAttrs, intAttrs, MQ_Q_NAME_LENGTH, qName,
          &CompCode, &Reason);
    if (CompCode != MQCC_OK) {
        printf("MQINQ failed with Condition code %d and Reason %d\n", CompCode, Reason);
        strcpy(qName, "unknown queue");
    }
    return;
}
```

Obrázek 82. Spravovaný odběratel MQ -část 2: tělo kódu.

```
W:\Subscribe2\Debug>solution2
Optional parameters: topicName, topicString
Values "STOCKS"  "IBM/PRICE"
Waiting 10 seconds for publications matching "NYSE/IBM/PRICE" from
"SYSTEM.MANAGED.NDURABLE.48A0AC7403300020    "
Received publication "150"
Completion code 0 and Return code 0

W:\Subscribe2\Debug>solution2 / NYSE/IBM/PRICE
Optional parameters: topicName, topicString
Values ""  "NYSE/IBM/PRICE"
Waiting 10 seconds for publications matching "NYSE/IBM/PRICE" from
"SYSTEM.MANAGED.NDURABLE.48A0AC7403310020    "
Received publication "150"
Completion code 0 and Return code 0
```

Obrázek 83. Odběratel MQ

V tomto příkladu jsou k dispozici některé další komentáře k provedení kódu.

strncpy(sd.ObjectName, topicName, MQ_Q_NAME_LENGTH);
Má-li parametr topicName hodnotu null nebo je prázdný *(výchozí hodnota)*, nebude název tématu použit k výpočtu vyřešeného řetězce tématu.

sd.ObjectString.VSPtr = topicString;
Místo použití pouze předdefinovaného objektu tématu v tomto příkladu poskytuje programátor objekt tématu a řetězec tématu, který je zkombinován s použitím produktu MQSUB. Všimněte si, že řetězec tématu má strukturu MQCHARV .

sd.ObjectString.VSLength = MQVS_NULL_TERMINATED;
Alternativa k nastavení délky pole MQCHARV .

sd.Options = MQSO_CREATE | MQSO_MANAGED | MQSO_NON_DURABLE | MQSO_FAIL_IF_QUIESCING;
Po definování řetězce tématu potřebují příznaky produktu sd.Options nejpečlivější pozornost. Existuje mnoho voleb, příklad uvádí pouze ty, které jsou nejčastěji používané. Ostatní volby používají výchozí hodnoty.

1. Vzhledem k tomu, že odběr je *netrvalý*, znamená to, že má v aplikaci dobu životnosti otevřeného odběru, nastavte příznak MQSO_CREATE . Chcete-li čitelnost nastavit, můžete také nastavit příznak *(výchozí)* MQSO_NON_DURABLE .
2. Komplementování MQSO_CREATE je MQSO_RESUME. Oba příznaky lze nastavit společně; správce front buď vytvoří nový odběr, nebo obnoví existující odběr, podle toho, co je vhodné. Pokud však zadáte MQSO_RESUME , musíte také inicializovat strukturu MQCHARV pro sd.SubName, i když není odběr k obnovení. Selhání inicializace SubName vede k návratovému kódu 2440: MQRC_SUB_NAME_ERROR z MQSUB.

   Poznámka: MQSO_RESUME je vždy ignorován pro netrvalý spravovaný odběr: ale jeho uvedení bez inicializace struktury MQCHARV pro sd.SubName způsobí chybu.
3. Kromě toho existuje třetí příznak, který ovlivňuje způsob, jakým je odběr otevřen, MQSO_ALTER. Vzhledem ke správným oprávněním se vlastnosti obnovených odběrů mění tak, aby odpovídaly jiným atributům uvedeným v produktu MQSUB.

   Poznámka: Musí být zadán alespoň jeden z parametrů MQSO_CREATE, MQSO_RESUME a MQSO_ALTER . Viz Volby (MQLONG). Existují příklady použití všech tří příznaků v produktu “Příklad 3: Nespravovaný odběratel MQ” na stránce 790.
4. Nastavte produkt MQSO_MANAGED pro správce front tak, aby byl pro vás automaticky určen odběr.

sd.ObjectString.VSLength = MQVS_NULL_TERMINATED;
Volitelně vynechte nastavení délky MQCHARV pro prázdné řetězce s hodnotou null a místo toho použijte příznak ukončení znaku null.

sd.ResObjectString.VSPtr = resTopicStr;
Výsledný řetězec tématu se vypisuje v prvním souboru printf v programu. Nastavte parametr MQCHARV ResObjectString for IBM MQ tak, aby vracel přeložený řetězec zpět do programu.

Poznámka: Příkaz resTopicStringBuffer je inicializován na hodnoty null v souboru memset(resTopicStr, 0, sizeof(resTopicStrBuffer)). Vrácené řetězce témat nekončí koncovým znakem null.

sd.ResObjectString.VSBufSize = sizeof(resTopicStrBuffer)-1;
Nastavte velikost vyrovnávací paměti sd.ResObjectString na hodnotu menší, než je její skutečná velikost. To zabraňuje přepsání ukončovače null, který je poskytnut, v případě, že vyřešený řetězec tématu vyplní celou vyrovnávací paměť.

Poznámka: Pokud je řetězec tématu delší než sizeof(resTopicStrBuffer)-1, není vrácena žádná chyba. I když VSLength > VSBufSiz délka vrácená v sd.ResObjectString.VSLength je délka celého řetězce a nemusí být nutně délka vráceného řetězce. Testujte sd.ResObjectString.VSLength < sd.ResObjectString.VSBufSiz a potvrďte, že řetězec tématu je dokončen.

MQSUB(Hconn, &sd, &Hobj, &Hsub, &CompCode, &Reason);
Funkce MQSUB vytvoří odběr. Pokud je netrvanlivý, pravděpodobně se o její název nezajímáte, ačkoli můžete zkontrolovat jeho stav v Průzkumníku IBM MQ . Jako vstup můžete zadat parametr sd.SubName , takže víte, jak se má hledat; zřejmě se musíte vyvarovat kolizi názvů s jinými odběry.

MQCLOSE(Hconn, &Hsub, MQCO_REMOVE_SUB, &CompCode, &Reason);
Zavírání odběru a fronty odběru je volitelné. V tomto příkladu je odběr uzavřen, ale ne ve frontě. Volba MQCLOSE MQCO_REMOVE_SUB je standardně v tomto případě, protože odběr je netrvalý. Použití MQCO_KEEP_SUB je chyba.

Poznámka: odběr *fronta* není uzavřen produktem MQSUBa jeho popisovač Hobjzůstává platný do doby, než je fronta uzavřena produktem MQCLOSE nebo MQDISC. Dojde-li k předčasnému ukončení aplikace, dojde k vyčištění fronty a odběru někdy po ukončení aplikace správcem front.

Související pojmy

“Příklad 1: Spotřebitel publikování MQ” na stránce 783
Spotřebitel zpráv produktu MQ je spotřebitel zpráv produktu IBM MQ , který se nepřihlašují k odběru vlastních témat.

“Příklad 3: Nespravovaný odběratel MQ” na stránce 790
Nespravovaný odběratel je důležitou třídou aplikace odběratele. S ní kombinujete výhody publikování/odběru s *ovládacím prvkem* řazení do fronty a spotřebou publikací. Tento příklad ukazuje různé způsoby kombinování odběrů a front.

“Zapisování aplikací vydavatele” na stránce 775
Začněte pracovat se zápisem aplikací vydavatele tak, že prostudujete dva příklady. První je modelován co nejpřesněji v okamžiku, kdy aplikace boduje aplikaci do fronty, a druhý demonstruje vytváření témat dynamicky-běžnější vzor pro aplikace vydavatele.

Příklad 3: Nespravovaný odběratel MQ
Nespravovaný odběratel je důležitou třídou aplikace odběratele. S ní kombinujete výhody publikování/odběru s *ovládacím prvkem* řazení do fronty a spotřebou publikací. Tento příklad ukazuje různé způsoby kombinování odběrů a front.

Nespravovaný vzor je častěji přidružen k odběrům *trvalých* než *netrvalých*. Životní cyklus odběru vytvořeného nespravovaným odběratelem je obvykle nezávislý na životním cyklu samotné odběratelské aplikace. Díky tomu, že odběr bude trvalý, obdrží publikování publikování, i když není aktivní žádná odebírající aplikace.

Můžete vytvořit trvalé odběry *spravované* , chcete-li dosáhnout stejného výsledku, ale některé aplikace vyžadují větší flexibilitu a kontrolu nad frontami a zprávami, než je možné u spravovaného odběru. U trvalého spravovaného odběru vytvoří správce front trvalou frontu pro publikování, která se shodují s tématem odběru. Odstraní frontu a přidružená publikování po odstranění odběru.

Obvykle jsou použity trvalé *spravované* odběry, je-li životní cyklus aplikace a odběr v podstatě stejný, ale je těžké zaručit záruku. Po vytvoření trvalého odběru a použití sdílených odběrů každá aplikace, která sdílí odběr, otevře stejnou spravovanou frontu a získává zprávy z ní.

Spravovaný odběr je takový, kde produkt IBM MQ zpracovává odběr a provádí registraci a deregistrování pro vás, zatímco v rámci *nespravovaného* odběru je aplikace odpovědná za určení fronty, kde jsou odběry uloženy.

Správce front implicitně otevře trvalou spravovanou frontu odběru pro odběratele takovým způsobem, že sdílené zpracování fronty není možné. Kromě toho nelze pro každou spravovanou frontu vytvořit více než jeden odběr a fronty lze lépe spravovat, protože máte menší kontrolu nad názvy front. Z těchto důvodů zvažte, zda je odběratel *nespravováno* MQ vhodnější pro aplikace vyžadující trvalé odběry než odběratel produktu *spravované* MQ .

Kód v produktu Obrázek 86 na stránce 795 demonstruje nespravovaný vzorek trvalého odběru. Pro ilustraci tento kód také vytváří nespravované, netrvalé odběry. Tento příklad ilustruje následující fazety vzorku:

- Na odběrech poptávky: řetězce témat odběru jsou dynamické. Poskytují ji aplikace, když je spuštěna.

- Zjednodušená správa témat odběru: Správa témat odběru je zjednodušena definováním kořenové části řetězce tématu odběru pomocí administrativně definovaného tématu. Tím skryjete kořenovou část stromu témat z aplikace. Skrytím kořenové části může být odběratel implementován do různých stromů témat.
- Flexibilní správa odběrů: Můžete definovat odběr buď administrativně, nebo jej vytvořit na vyžádání v programu odběratele. Mezi administrativně a programově vytvořenými odběry neexistuje rozdíl mezi administrativně a programově vytvořenými odběry, s výjimkou atributu, který ukazuje, jak byl vytvořen odběr. Existuje třetí typ odběru, který je vytvořen automaticky správcem front pro distribuci odběrů. Všechny odběry se zobrazí v Průzkumníku IBM MQ .
- Flexibilní přidružení odběrů s frontami: Předdefinovaná lokální fronta je přidružena k odběru funkcí MQSUB . Existují různé způsoby použití funkce MQSUB pro přidružení odběrů s frontami:
  - Přidružte odběr ke frontě, která má *ne* existující odběry, `MQSO_CREATE + (Hobj from MQOPEN)`.
  - Přidružte *nový* odběr ke frontě s existujícími odběry `MQSO_CREATE + (Hobj from MQOPEN)`.
  - Přesuňte existující odběr do jiné fronty, `MQSO_ALTER + (Hobj from MQOPEN)`.
  - Obnovte existující odběr přidružený k existující frontě, `MQSO_RESUME + (Hobj = MQHO_NONE)` nebo `MQSO_RESUME + (Hobj = from MQOPEN of queue with existing subscription)`.
  - Kombinací produktu `MQSO_CREATE | MQSO_RESUME | MQSO_ALTER` v různých kombinacích můžete obsloužit různé vstupní stavy odběru a fronty, aniž byste museli kódovat více verzí produktu MQSUB s různými hodnotami `sd.Options` .
  - Případně kódováním specifické volby produktu `MQSO_CREATE | MQSO_RESUME | MQSO_ALTER` vrátí správce front chybu (Tabulka 109 na stránce 792). jsou-li stavy odběru a fronty poskytnuté jako vstup do produktu MQSUB nekonzistentní s hodnotou `sd.Options`. Příkaz Obrázek 92 na stránce 798 zobrazuje výsledky vydání MQSUB pro odběr X s různými individuálními nastaveními parametru `sd.Options` a předáním tří různých manipulátorů objektů.

  Prozkoumejte různé vstupy do vzorového programu v produktu Obrázek 85 na stránce 794 a obeznamte se s těmito různými druhy chyb. Jedna běžná chyba, `RC = 2440`, která není zahrnuta v případech uvedených v tabulce, je chyba názvu odběru. je obvykle způsoben předáním hodnoty null nebo neplatnému názvu odběru s `MQSO_RESUME` nebo `MQSO_ALTER`.
- Multiprocessing: Můžete sdílet mezi mnoha spotřebiteli práci na čtení publikací. Všechny publikace se nacházejí v jedné frontě přidružené k odběru tématu odběru. Spotřebitelé mají možnost volby otevření fronty přímo pomocí produktu MQOPEN nebo obnovení odběru pomocí produktu MQSUB.
- Spojení odběru: Více odběrů lze vytvořit ve stejné frontě. Buďte opatrní s touto schopností, protože to může vést k překrývajícím se odběrům a přijímat stejné publikování vícekrát. Volba `MQSO_GROUP_SUB` eliminuje duplicitní publikování v důsledku překrývajících se odběrů.
- Subscriber and consumer separation: stejně jako tři modely zákazníků ilustrované v příkladech je dalším modelem oddělit odběratele od odběratele. Jedná se o variantu nespravovaného odběratele MQ Subscriber, ale spíše než vydání MQOPEN a MQSUB ve stejném programu, jeden program se přihlásí k odběru publikací a jiný program je spotřebovává. Odběratel může být například součástí klastru publikování/odběru a konzumenta připojený ke správci front mimo klastr správců front. Konzument přijímá publikace prostřednictvím standardního distribuovaného řazení do fronty tím, že definuje frontu odběru jako definici vzdálené fronty.

Porozumění chování produktu `MQSO_CREATE | MQSO_RESUME | MQSO_ALTER` je důležité, zvláště pokud plánujete zjednodušit svůj kód pomocí kombinací těchto voleb. Prostudujte si tabulku Tabulka 109 na stránce 792 , která uvádí výsledky předávání různých manipulátorů front příkazu MQSUB, a výsledky spuštění vzorového programu zobrazeného v souboru Obrázek 87 na stránce 796 na hodnotu Obrázek 92 na stránce 798.

Scénář použitý ke konstrukci tabulky má jeden odběr X a dvě fronty, A a B. Parametr názvu odběru `sd.SubName` je nastaven na hodnotu X, název odběru připojený ke frontě A. K frontě B není připojen žádný odběr.

V produktu Tabulka 109 na stránce 792je MQSUB předán odběr X a popisovač fronty do fronty A. Výsledky voleb odběru jsou následující:

- MQSO_CREATE selže, protože manipulační prostředek fronty odpovídá frontě A , která již má odběr produktu X. Porovnejte toto chování s úspěšným voláním. Toto volání je úspěšné, protože fronta B nemá k sobě přiložený odběr X .
- MQSO_RESUME je úspěšný, protože manipulátor fronty odpovídá frontě A , která již má odběr produktu X. Naopak volání selže, pokud odběr X ve frontě Aneexistuje.
- Produkt MQSO_ALTER se chová podobným způsobem jako MQSO_RESUME s ohledem na otevření odběru a fronty. Pokud však atributy obsažené v deskriptoru odběru předané produktu MQSUB se liší od atributů odběru, MQSO_RESUME se nezdaří, zatímco MQSO_ALTER uspěje, pokud má instance programu oprávnění k pozměnění atributů. Všimněte si, že řetězec tématu nelze nikdy změnit v odběru, ale spíše než vrátit chybu, produkt MQSUB ignoruje hodnoty názvu tématu a řetězce tématu v deskriptoru odběru a použije hodnoty v existujícím odběru.

Dále se podívejte na Tabulka 109 na stránce 792 , kde MQSUB prošel odběrem X a popisovač fronty do fronty B. Výsledky voleb odběru jsou následující:

- Produkt MQSO_CREATE uspěje a vytvoří odběr X ve frontě B , protože se jedná o nový odběr ve frontě B.
- MQSO_RESUME selže. Příkaz MQSUB hledá odběr X ve frontě B a nenalezne jej, ale nevrací *RC = 2428-odběr X neexistuje*, vrací *RC = 2019-Fronta odběru neodpovídá manipulátoru objektu fronty*. Chování třetí volby MQSO_ALTER napovídá o příčině této neočekávané chyby. MQSUB očekává, že se popisovač fronty bude odkazovat na frontu s odběrem. Před kontrolou, zda existuje odběr uvedený v produktu sd . SubName , je nejprve tato kontrola provedena.
- MQSO_ALTER uspěje a přesune odběr z fronty A do fronty B.

Případ, který není zobrazen v tabulce, je případ, kdy název odběru odběru ve frontě A neodpovídá názvu odběru v produktu sd . SubName. Toto volání selhává s *RC = 2428-odběr X neexistuje ve frontě A*.

Tabulka 109. Chyby z MQSUB s různými manipulátory front a kombinace odběrů

	Fronta A **Odběr X** **Fronta B** **Bez odběru**	**Fronta A** **Bez odběru** **Fronta B** **Bez odběru**
Objekt Hobj pro frontu *Queue A* předaný funkci MQSUB	**MQSO_CREATE** RC = 2432-Odběr X již ve frontě A existuje **MQSO_RESUME** Obnoví odběr X ve frontě A **MQSO_ALTER** Obnoví odběr X ve frontě A a povolí změny	**MQSO_CREATE** Vytvoří odběr X ve frontě A **MQSO_RESUME** RC = 2428-Odběr X ve frontě A neexistuje **MQSO_ALTER** RC = 2428-Odběr X ve frontě A neexistuje
Objekt Hobj pro frontu *Queue B* předaný funkci MQSUB	**MQSO_CREATE** Vytvoří nový odběr X ve frontě B **MQSO_RESUME** RC = 2019-Fronta odběru se neshoduje s manipulátorem objektu fronty **MQSO_ALTER** Přesunout odběr X z fronty A do fronty B	**MQSO_CREATE** Vytvoří nový odběr X ve frontě B **MQSO_RESUME** RC = 2428-odběr X na frontě B neexistuje **MQSO_ALTER** RC = 2428-odběr X na frontě B neexistuje

Tabulka 109. Chyby z MQSUB s různými manipulátory front a kombinace odběrů (pokračování)		
	Fronta A **Odběr X** **Fronta B** **Bez odběru**	**Fronta A** **Bez odběru** **Fronta B** **Bez odběru**
MQHO_NONE předán rutině MQSUB	**MQSO_CREATE** RC = 2019-Špatný popisovač objektu: nastaví příznak MQSO_MANAGED pro vytvoření spravovaného odběru a vytvoření spravované fronty **MQSO_RESUME** Obnoví odběr X ve frontě A a vrátí objekt Hobj do fronty A **MQSO_ALTER** Obnoví odběr X ve frontě A, vrátí objekt Hobj do fronty A a povolí změny.	**MQSO_CREATE** RC = 2019-Špatný popisovač objektu: nastaví příznak MQSO_MANAGED pro vytvoření spravovaného odběru a vytvoření spravované fronty **MQSO_RESUME** RC = 2428-Žádné předplatné X **MQSO_ALTER** RC = 2019-Špatný popisovač objektu: Žádná fronta A nebo B

Poznámka: Styl kompaktního kódování je určen pro přehlednost, nikoli pro použití v produkčním prostředí.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <cmqc.h>

void inquireQname(MQHCONN HConn, MQHOBJ Hobj, MQCHAR48 qName);

int main(int argc, char **argv)
{
    MQCHAR48 topicNameDefault            = "STOCKS";
    char     topicStringDefault[]        = "IBM/PRICE";
    char     subscriptionNameDefault[]   = "IBMSTOCKPRICESUB";
    char     subscriptionQueueDefault[]  = "STOCKTICKER";
    char     publicationBuffer[101];      /* Allocate to receive messages     */
    char     resTopicStrBuffer[151];      /* Allocate to resolve topic string */
    MQCHAR48 qmName = "";                 /* Default queue manager            */
    MQCHAR48 qName = "";                  /* Allocate storage for MQINQ       */

    MQHCONN  Hconn = MQHC_UNUSABLE_HCONN; /* connection handle                */
    MQHOBJ   Hobj = MQHO_NONE;            /* subscription queue handle        */
    MQHOBJ   Hsub = MQSO_NONE;            /* subscription handle              */
    MQLONG   CompCode = MQCC_OK;          /* completion code                  */
    MQLONG   Reason = MQRC_NONE;          /* reason code                      */
    MQLONG   messlen = 0;
    MQOD     od = {MQOD_DEFAULT};         /* Unmanaged subscription queue     */
    MQSD     sd = {MQSD_DEFAULT};         /* Subscription Descriptor          */
    MQMD     md = {MQMD_DEFAULT};         /* Message Descriptor               */
    MQGMO   gmo = {MQGMO_DEFAULT};        /* get message options              */
    MQLONG   sdOptions = MQSO_CREATE | MQSO_RESUME | MQSO_DURABLE |
MQSO_FAIL_IF_QUIESCING;

    char *   topicName = topicNameDefault;
    char *   topicString = topicStringDefault;
    char *   subscriptionName =  subscriptionNameDefault;
    char *   subscriptionQueue = subscriptionQueueDefault;
    char *   publication = publicationBuffer;
    char *   resTopicStr = resTopicStrBuffer;
    memset(resTopicStrBuffer, 0, sizeof(resTopicStrBuffer));
```

Obrázek 84. Nespravovaný odběratel MQ -část 1: deklarace.

```
    switch(argc){                   /* Replace defaults with args if provided */
default:
        switch((argv[5][0])) {
case('A'):  sdOptions =  MQSO_ALTER  | MQSO_DURABLE | MQSO_FAIL_IF_QUIESCING;
            break;
case('C'):  sdOptions =  MQSO_CREATE | MQSO_DURABLE | MQSO_FAIL_IF_QUIESCING;
            break;
case('R'):  sdOptions =  MQSO_RESUME | MQSO_DURABLE | MQSO_FAIL_IF_QUIESCING;
            break;
default:    ;
        }
case(5):
    if (strcmp(argv[4],"/")) /* "/" invalid = No subscription  */
        subscriptionQueue = argv[4];
    else {
        *subscriptionQueue = '\0';
        if (argc > 5) {
            if (argv[5][0] == 'C') {
                sdOptions = sdOptions + MQSO_MANAGED;
            }
        }
        else
            sdOptions = sdOptions + MQSO_MANAGED;
    }

case(4):
    if (strcmp(argv[3],"/")) /* "/" invalid = No subscription  */
        subscriptionName = argv[3];
    else {
        *subscriptionName = '\0';
        sdOptions = sdOptions - MQSO_DURABLE;
    }
case(3):
    if (strcmp(argv[2],"/")) /* "/" invalid = No topic string  */
        topicString = argv[2];
    else
        *topicString = '\0';
case(2):
    if (strcmp(argv[1],"/")) /* "/" invalid = No topic object  */
        topicName = argv[1];
    else
        *topicName = '\0';
case(1):
    sd.Options = sdOptions;
    printf("Optional parameters: "
    printf("topicName, topicString, subscriptionName, subscriptionQueue, A(lter)|C(reate)|
R(esume)\n");
    printf("Values \"%-.48s\"  \"%s\" \"%s\"  \"%-.48s\" sd.Options=%d\n",
           topicName, topicString, subscriptionName, subscriptionQueue, sd.Options);
}
```

Obrázek 85. Nespravovaný odběratel MQ -část 2: manipulace s parametry.

Další komentáře týkající se zacházení s parametry v tomto příkladu jsou následující:

switch((argv[5][0]))
Máte možnost volby A lter | C reate | R esume v parametru 5, abyste otestovali účinek přepisující části nastavení parametru MQSUB , který je v příkladu použit jako výchozí. Výchozí nastavení použité v příkladu je MQSO_CREATE | MQSO_RESUME | MQSO_DURABLE.

Poznámka: Nastavení MQSO_ALTER nebo MQSO_RESUME bez nastavení MQSO_DURABLE je chyba a sd.SubName musí být nastaven a odkazovat na odběr, který může být obnoven nebo změněn.

***subscriptionQueue = '\0';**
sdOptions = sdOptions + MQSO_MANAGED;
Je-li výchozí fronta odběru, STOCKTICKER je nahrazena řetězcem s hodnotou null, pokud je nastaven parametr MQSO_CREATE , tento příklad nastaví příznak MQSO_MANAGED a vytvoří frontu dynamického odběru. Je-li Alter or Resume nastaveno v pátém parametru, chování příkladu bude záviset na hodnotě subscriptionName.

***subscriptionName = '\0';**
sdOptions = sdOptions - MQSO_DURABLE;
Je-li standardní odběr IBMSTOCKPRICESUBnahrazen řetězcem s hodnotou null, odebere tento příkaz příznak MQSO_DURABLE . Spustíte-li příklad poskytující výchozí hodnoty pro ostatní parametry, vytvoří se další dočasný odběr určený pro STOCKTICKER a přijme duplicitní publikace. Při příštím spuštění tohoto příkladu, bez parametrů, obdržíte znovu pouze jednu publikaci.

```
    do {
        MQCONN(qmName, &Hconn, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        if (strlen(subscriptionQueue)) {
            strncpy(od.ObjectName, subscriptionQueue, MQ_Q_NAME_LENGTH);
            MQOPEN(Hconn, &od, MQOO_INPUT_AS_Q_DEF | MQOO_FAIL_IF_QUIESCING | MQOO_INQUIRE,
                  &Hobj, &CompCode, &Reason);
            if (CompCode != MQCC_OK) break;
        }
        strncpy(sd.ObjectName, topicName, MQ_TOPIC_NAME_LENGTH);
        sd.ObjectString.VSPtr = topicString;
        sd.ObjectString.VSLength = MQVS_NULL_TERMINATED;
        sd.SubName.VSPtr = subscriptionName;
        sd.SubName.VSLength = MQVS_NULL_TERMINATED;
        sd.ResObjectString.VSPtr = resTopicStr;
        sd.ResObjectString.VSBufSize = sizeof(resTopicStrBuffer)-1;
        MQSUB(Hconn, &sd, &Hobj, &Hsub, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        gmo.Options = MQGMO_WAIT | MQGMO_NO_SYNCPOINT | MQGMO_CONVERT;
        gmo.WaitInterval = 10000;
        gmo.MatchOptions = MQMO_MATCH_CORREL_ID;
        memcpy(md.CorrelId, sd.SubCorrelId, MQ_CORREL_ID_LENGTH);
        inquireQname(Hconn, Hobj, qName);
        printf("Waiting %d seconds for publications matching \"%s\" from %-0.48s\n",
               gmo.WaitInterval/1000, resTopicStr, qName);
        do {
            memcpy(md.MsgId, MQMI_NONE, sizeof(md.MsgId));
            memcpy(md.CorrelId, MQCI_NONE, sizeof(md.CorrelId));
            md.Encoding       = MQENC_NATIVE;
            md.CodedCharSetId = MQCCSI_Q_MGR;
            MQGET(Hconn, Hobj, &md, &gmo, sizeof(publication), publication, &messlen,
&CompCode, &Reason);
            if (Reason == MQRC_NONE)
                printf("Received publication \"%s\"\n", publication);
        }
        while (CompCode == MQCC_OK);
        if (CompCode != MQCC_OK && Reason != MQRC_NO_MSG_AVAILABLE) break;
        MQCLOSE(Hconn, &Hsub, MQCO_NONE, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        MQCLOSE(Hconn, &Hobj, MQCO_NONE, &CompCode, &Reason);
        if (CompCode != MQCC_OK) break;
        MQDISC(&Hconn, &CompCode, &Reason);
    } while (0);
    printf("Completion code %d and Return code %d\n", CompCode, Reason);
}
void inquireQname(MQHCONN Hconn, MQHOBJ Hobj, MQCHAR48 qName) {
#define _selectors 1
#define _intAttrs 1

    MQLONG select[_selectors] = {MQCA_Q_NAME}; /* Array of attribute selectors    */
    MQLONG intAttrs[_intAttrs];            /* Array of integer attributes           */
    MQLONG CompCode, Reason;
    MQINQ(Hconn, Hobj, _selectors, select, _intAttrs, intAttrs, MQ_Q_NAME_LENGTH, qName,
&CompCode, &Reason);
    if (CompCode != MQCC_OK) {
        printf("MQINQ failed with Condition code %d and Reason %d\n", CompCode, Reason);
        strncpy(qName, "unknown queue", MQ_Q_NAME_LENGTH);
    }
    return;
}
```

Obrázek 86. Nespravovaný odběratel MQ -část 3: tělo kódu.

Další komentáře k kódu v tomto příkladu jsou následující:

if (strlen(subscriptionQueue))
Pokud neexistuje žádný název fronty odběru, pak příklad používá MQHO_NONE jako hodnotu Hobj.

MQOPEN(...);
Je otevřena fronta odběrů a popisovač fronty uložen v produktu Hobj.

MQSUB(Hconn, &sd, &Hobj, &Hsub, &CompCode, &Reason);
Odběr se otevře pomocí produktu Hobj , který prošel produktem MQOPEN (nebo MQHO_NONE , pokud neexistuje žádný název fronty odběru). Nespravovaná fronta může být obnovena bez explicitního otevření s použitím MQOPEN.

MQCLOSE(Hconn, &Hsub, MQCO_NONE, &CompCode, &Reason);
Odběr je uzavřen pomocí popisovače odběru. V závislosti na tom, zda je odběr trvalý či nikoli, je odběr uzavřen s implicitním produktem MQCO_KEEP_SUB nebo MQCO_REMOVE_SUB. Trvalý odběr je možné zavřít s produktem MQCO_REMOVE_SUB, nelze však zavřít trvalý odběr s produktem MQCO_KEEP_SUB. Akce produktu MQCO_REMOVE_SUB odebere odběr, který zastaví jakékoli další publikace odesílané do fronty odběru.

MQCLOSE(Hconn, &Hobj, MQCO_NONE, &CompCode, &Reason);
Pokud je odběr nespravovaný, není provedena žádná speciální akce. Je-li fronta spravována a odběr je uzavřen explicitním nebo implicitním produktem MQCO_REMOVE_SUB, budou všechny publikace vymazány z fronty a fronty odstraněné v tomto bodu.

gmo.MatchOptions = MQMO_MATCH_CORREL_ID;
memcpy(md.CorrelId, sd.SubCorrelId, MQ_CORREL_ID_LENGTH);
Ujistěte se, že přijaté zprávy jsou ty, které jsou pro náš odběr.

Výsledky z příkladu ilustrují aspekty publikování/odběru:

V produktu Obrázek 87 na stránce 796 se příklad spustí publikováním 130 na téma NYSE/IBM/PRICE .

```
W:\Subscribe3\Debug>..\..\Publish2\Debug\publishstock
Provide parameters: TopicObject TopicString Publication
Publish "130" to topic "STOCKS" and topic string "IBM/PRICE"
Published "130" to topic string "NYSE/IBM/PRICE"
Completion code 0 and Return code 0
```

Obrázek 87. Publikovat 130 na NYSE/IBM/PRICE

V Obrázek 88 na stránce 796 provedení příkladu s použitím výchozích parametrů přijímá zachované publikování 130. Zadaný objekt tématu a řetězec tématu jsou ignorovány, jak je zobrazeno v tématu Obrázek 92 na stránce 798. Objekt tématu a řetězec tématu jsou vždy převzaty z objektu odběru, je-li zadán, a řetězec tématu je neměnný. Skutečné chování tohoto příkladu závisí na volbě nebo kombinaci MQSO_CREATE, MQSO_RESUMEa MQSO_ALTER. V tomto příkladě je vybrána volba MQSO_RESUME .

```
W:\Subscribe3\Debug>solution3
Optional parameters: topicName, topicString, subscriptionName, subscriptionQueue, A(lter)|
C(reate)|R(esume)
Values "STOCKS" "IBM/PRICE" "IBMSTOCKPRICESUB" "STOCKTICKER" sd.Options=8206
Waiting 10 seconds for publications matching "NYSE/IBM/PRICE" from STOCKTICKER
Received publication "130"
Completion code 0 and Return code 0
```

Obrázek 88. Přijmout zachované publikování

V (Obrázek 89 na stránce 797) nejsou přijata žádná publikování, protože trvalý odběr již obdržel zachované publikování. V tomto příkladu bude odběr obnoven tak, že zadáte pouze název odběru bez názvu fronty. Pokud byl zadán název fronty, byla by nejprve otevřena fronta a obsluha byla předána produktu MQSUB.

Poznámka: Chyba 2038 z MQINQ je způsobena implicitní MQOPEN z STOCKTICKER tím, že MQSUB nezahrnuje volbu MQOO_INQUIRE . Vyvarovat se návratového kódu 2038 z MQINQ tím, že otevřete frontu explicitně.

```
W:\Subscribe3\Debug>solution3 STOCKS IBM/PRICE IBMSTOCKPRICESUB / Resume
Optional parameters: topicName, topicString, subscriptionName, subscriptionQueue, A(lter)|
C(reate)|R(esume)
Values "STOCKS" "IBM/PRICE" "IBMSTOCKPRICESUB" "" sd.Options=8204
MQINQ failed with Condition code 2 and Reason 2038
Waiting 10 seconds for publications matching "NYSE/IBM/PRICE" from unknown queue
Completion code 0 and Return code 0
```

Obrázek 89. Obnovit odběr

V produktu Obrázek 90 na stránce 797vytváří příklad netrvalý nespravovaný odběr pomocí parametru STOCKTICKER jako místo určení. Vzhledem k tomu, že se jedná o nový odběr, obdrží zachované publikování.

```
W:\Subscribe3\Debug>solution3 STOCKS IBM/PRICE / STOCKTICKER Create
Optional parameters: topicName, topicString, subscriptionName, subscriptionQueue, A(lter)|
C(reate)|R(esume)
Values "STOCKS" "IBM/PRICE" "" "STOCKTICKER" sd.Options=8194
Waiting 10 seconds for publications matching "NYSE/IBM/PRICE" from STOCKTICKER
Received publication "130"
Completion code 0 and Return code 0
```

Obrázek 90. Přijmout zachované publikování s novým nespravovaným nestálým odběrem

Chcete-li demonstrovat překrývající se odběry v produktu Obrázek 91 na stránce 797, je odeslána jiná publikace, která mění zachované publikování. Dále je vytvořen nový netrvalý nespravovaný odběr, který neposkytuje název odběru. Zachované publikování je přijato dvakrát, jednou pro nový odběr, a jednou pro trvalý odběr IBMSTOCKPRICESUB , který je stále aktivní ve frontě STOCKTICKER . Příklad je obrázek, že se jedná o frontu, která má odběry, nikoli aplikaci. I když neodkazujete na odběr IBMSTOCKPRICESUB v tomto vyvolání aplikace, aplikace obdrží tuto publikaci dvakrát: jednou z trvalého odběru, který byl vytvořen administrativně, a jednou z netrvalého odběru vytvořeného aplikací samotnou.

```
W:\Subscribe3\Debug>..\..\Publish2\Debug\publishstock
Provide parameters: TopicObject TopicString Publication
Publish "130" to topic "STOCKS" and topic string "IBM/PRICE"
Published "130" to topic string "NYSE/IBM/PRICE"
Completion code 0 and Return code 0

W:\Subscribe3\Debug>solution3 STOCKS IBM/PRICE / STOCKTICKER Create
Optional parameters: topicName, topicString, subscriptionName, subscriptionQueue, A(lter)|
C(reate)|R(esume)
Values "STOCKS" "IBM/PRICE" "" "STOCKTICKER" sd.Options=8194
Waiting 10 seconds for publications matching "NYSE/IBM/PRICE" from STOCKTICKER
Received publication "130"
Received publication "130"
Completion code 0 and Return code 0
```

Obrázek 91. Překrývání odběrů

V produktu Obrázek 92 na stránce 798 tento příklad ukazuje, že zadání nového řetězce tématu a existujícího odběru nevedlo ke změně odběru.

1. V prvním případě produkt `Resume` obnoví existující odběr, jak byste mohli očekávat, a ignoruje změněný řetězec tématu.
2. Ve druhém případě, `Alter` způsobí chybu, `RC = 2510, Topic not alterable`.
3. Ve třetím příkladu příkaz `Create` způsobí chybu `RC = 2432, Sub already exists`.

```
W:\Subscribe3\Debug>solution3 "" NASDAC/IBM/PRICE IBMSTOCKPRICESUB STOCKTICKER Resume
Optional parameters: topicName, topicString, subscriptionName, subscriptionQueue, A(lter)|C(reate)|R(esume)
Values "" "NASDAC/IBM/PRICE" "IBMSTOCKPRICESUB" "STOCKTICKER" sd.Options=8204
Waiting 10 seconds for publications matching "NYSE/IBM/PRICE" from STOCKTICKER
Received publication "130"
Completion code 0 and Return code 0

W:\Subscribe3\Debug>solution3 "" NASDAC/IBM/PRICE IBMSTOCKPRICESUB STOCKTICKER Alter
Optional parameters: topicName, topicString, subscriptionName, subscriptionQueue, A(lter)|C(reate)|R(esume)
Values "" "NASDAC/IBM/PRICE" "IBMSTOCKPRICESUB" "STOCKTICKER" sd.Options=8201
Completion code 2 and Return code 2510

W:\Subscribe3\Debug>solution3 "" NASDAC/IBM/PRICE IBMSTOCKPRICESUB STOCKTICKER Create
Optional parameters: topicName, topicString, subscriptionName, subscriptionQueue, A(lter)|C(reate)|R(esume)
Values "" "NASDAC/IBM/PRICE" "IBMSTOCKPRICESUB" "STOCKTICKER" sd.Options=8202
Completion code 2 and Return code 2432
```

Obrázek 92. Témata odběru nelze změnit

Související pojmy

“Příklad 1: Spotřebitel publikování MQ” na stránce 783
Spotřebitel zpráv produktu MQ je spotřebitel zpráv produktu IBM MQ , který se nepřihlašují k odběru vlastních témat.

“Příklad 2: Spravovaný odběratel MQ” na stránce 785
Spravovaný odběratel produktu MQ je upřednostňovaným vzorem pro většinu aplikací odběratele. Příklad vyžaduje *ne* administrative definition of queues, topics or subscriptions.

“Zapisování aplikací vydavatele” na stránce 775
Začněte pracovat se zápisem aplikací vydavatele tak, že prostudujete dva příklady. První je modelován co nejpřesněji v okamžiku, kdy aplikace boduje aplikaci do fronty, a druhý demonstruje vytváření témat dynamicky-běžnější vzor pro aplikace vydavatele.

Životní cykly publikování/odběru

Zvažte životní cykly témat, odběrů, odběratelů, publikací, vydavatelů a front v aplikacích pro návrh aplikací publikování/odběru.

Životní cyklus objektu, jako např. odběr, začíná jeho vytvořením a končí jeho odstraněním. Může také obsahovat jiné stavy a změny, které prochází, jako je dočasné pozastavení, které má nadřízená a podřízená témata, vypršení platnosti a odstranění.

Tradičně jsou objekty produktu IBM MQ , jako jsou fronty, vytvořeny administrativně nebo prostřednictvím administrativních programů s použitím Programmable Command Format (PCF). Publikování/odběr se liší v poskytování příkazových slov MQSUB a MQCLOSE k vytváření a odstraňování odběrů, které mají koncept spravovaných odběrů, které nejen vytvářejí a odstraňují fronty, ale také čistí nespotřebované zprávy a mají přidružení mezi administrativně vytvořenými objekty tématu a programově nebo administrativně vytvořeným řetězem témat.

Tento funkční bohatost zajišťuje širokou škálu požadavků na publikování/odběr a také zjednodušuje návrh některých běžných vzorců aplikace publikování/odběru. Spravované odběry například zjednodušují programování i administraci odběru, který je určen pouze tak dlouho jako program, který jej vytvořil. Nespravované odběry zjednodušují programování tam, kde dochází k uvolnění připojení mezi odebírajícími a odběratelskými publikacemi. Centrálně vytvořené odběry jsou užitečné v případech, kdy je vzorek jedním ze směrování přenosu publikování na spotřebitele na základě centralizovaného modelu řízení, například odesílání informací o letu do automatizovaných bran, zatímco programově vytvořené odběry mohou být použity, jsou-li zaměstnanci odpovědní za přihlášení k tomuto letu odpovědní za přihlášení k odběru pro daný let, zadáním čísla letu na bránu.

V tomto posledním příkladu může být spravovaný trvalý odběr vhodný: spravovaný, protože odběry jsou vytvářeny velmi často a mají jasný koncový bod při zavření brány a odběr může být programově odebrán; trvalý, aby nedošlo ke ztrátě záznamu o cestujících kvůli tomu, že program odběratele brány má nějaký důvod nebo jiný důvod.[9]. Chcete-li zahájit zveřejnění záznamů cestujících k bráně, možný návrh by byl pro použití brány pro obě přihlášení k odběru na osobní záznamy pomocí čísla brány, a publikovat události otevření brány s použitím čísla brány. Vydavatel odpovídá na událost otevření brány publikováním záznamů o cestujících-což by pak mohlo jít také do dalších zainteresovaných stran, jako je vyúčtování, na záznam letu, a na služby zákazníkům, na textová upozornění na mobilní telefony cestujících k bráně číslo.

[9] Vydavatel musí odeslat osobní záznamy jako trvalé zprávy, aby se zabránilo dalším možným poruchám, samozřejmě.

Centrálně spravovaný odběr může použít trvalý nespravovaný model, směrovat seznamy cestujících k bráně pomocí předem definované fronty pro každou bránu.

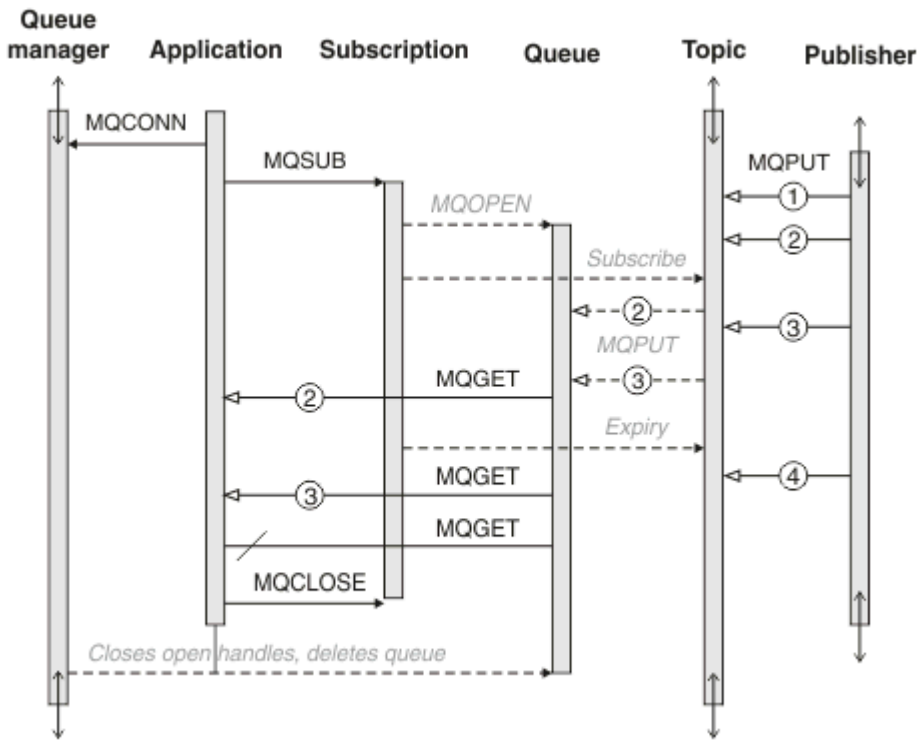
Následující tři příklady životních cyklů publikování/odběru ilustrují způsob interakce spravovaného netrvalého, spravovaného trvalého a nespravovaného trvalého odběratele s odběry, tématy, frontami, vydavateli a správcem front a jak mohou být odpovědnosti rozděleny mezi administrací a programy odběratele.

Spravovaný netrvalý odběratel

Produkt Obrázek 93 na stránce 799 zobrazuje aplikaci vytvářející spravovaný netrvalý odběr, získávání dvou zpráv, které jsou publikovány k tématu identifikovanému v odběru, a ukončování. Interakce označené šedým kurzívou s tečkovými šipkami jsou implicitní.

Je třeba poznamenat, že existují určité body.

1. Aplikace vytvoří odběr na téma, které již bylo publikováno dvakrát. Když odběratel obdrží svou první publikaci, obdrží *druhé* publikování, které je aktuálně zachovaným publikováním.
2. Správce front vytvoří dočasnou frontu odběru a vytvoří odběr pro dané téma.
3. Odběr má vypršení platnosti. Po vypršení platnosti odběru nejsou k tomuto odběru odeslány žádné další publikace, ale odběratel bude nadále dostávat zprávy publikované před tím, než vyprší platnost odběru. Vypršení platnosti publikování není ovlivněno vypršením platnosti odběru.
4. Čtvrtá publikace není umístěna ve frontě odběru a v důsledku toho poslední MQGET tuto publikaci nevrací.
5. Přestože odběratel zavře svůj odběr, nezavře své připojení k frontě nebo správci front.
6. Správce front se vyčistí krátce po ukončení aplikace. Vzhledem k tomu, že odběr je spravovaný a netrvalý, je odstraněna fronta odběru.



Obrázek 93. Spravované netrvalé odběratele lifelines

Spravovaný trvalý odběratel

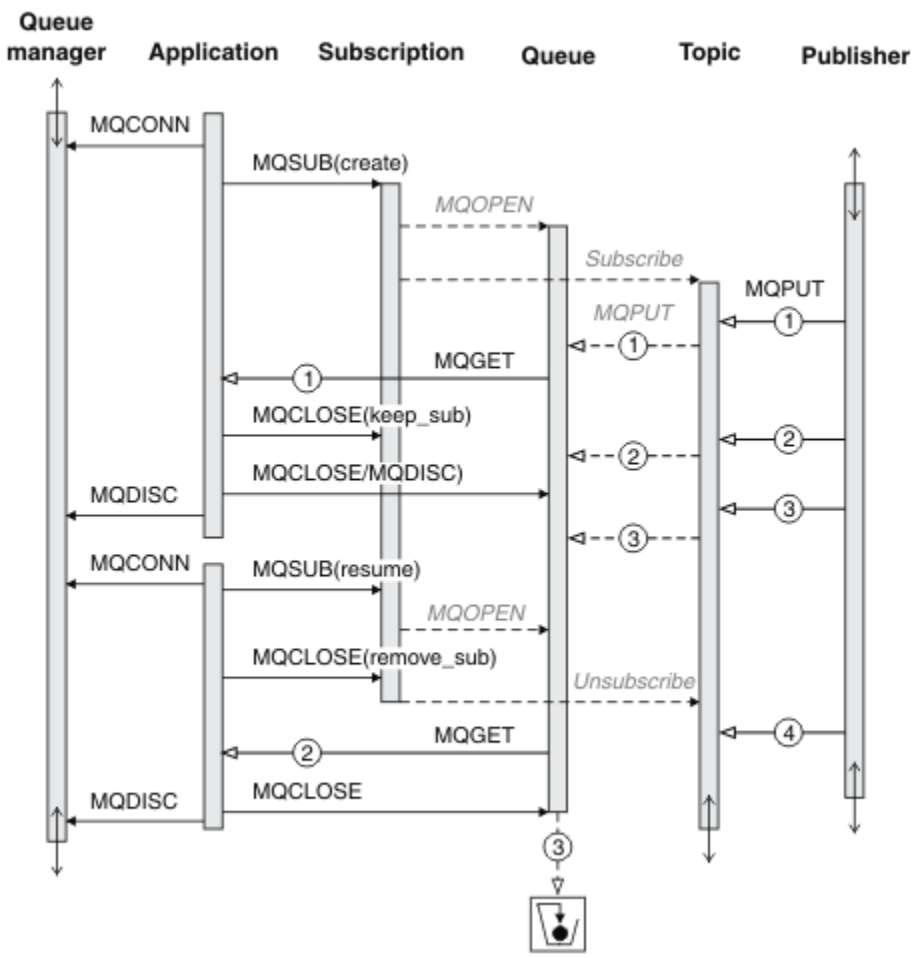
Spravovaný trvalý odběratel převezme předchozí krok dále a zobrazí spravovaný odběr, který přeživuje ukončení a restartování odběratelské aplikace.

Je zde několik nových bodů k poznámce.

1. V tomto příkladu, na rozdíl od posledního, téma publikování neexistovalo dříve, než bylo definováno v odběru.
2. Když se odběratel ukončí poprvé, zavře odběr s volbou MQCO_KEEP_SUB. To je výchozí chování pro implicitní zavření spravovaného trvalého odběru.
3. Když odběratel obnoví odběr, je znovu otevřena fronta odběrů.
4. Nová publikace 2, umístěná ve frontě před jejím opětným otevřením, je k dispozici pro MQGETi po odebrání odběru.

   Přestože je odběr trvalý, odběratel spolehlivě přijímá všechny zprávy odeslané vydavatelem pouze v případě, že *obojí* je trvalý a trvalý. Perzistence zpráv závisí na nastavení pole Persistent v souboru MQMD zprávy odeslaného vydavatelem. Odběratel nemá nad tím žádnou kontrolu.
5. Zavřením odběru s příznakem MQCO_REMOVE_SUB dojde k odebrání odběru a zastavování všech dalších publikování, která jsou umístěna ve frontě odběru. Po zavření fronty odběru odebere správce front nepřečtenou publikaci 3a poté ji odstraní. Akce je ekvivalentní administrativnímu odstranění odběru.

   Poznámka: Neodstraňujte frontu ručně, nebo zadejte příkaz MQCLOSE s volbou MQCO_DELETEnebo MQCO_PURGE_DELETE. Viditelné podrobnosti implementace spravovaného odběru nejsou součástí podporovaného rozhraní produktu IBM MQ . Správa správce front nemůže spolehlivě spravovat odběr, pokud nemá úplnou kontrolu.



Obrázek 94. Lifeliny spravovaného trvalého odběratele

Nespravovaný trvalý odběratel

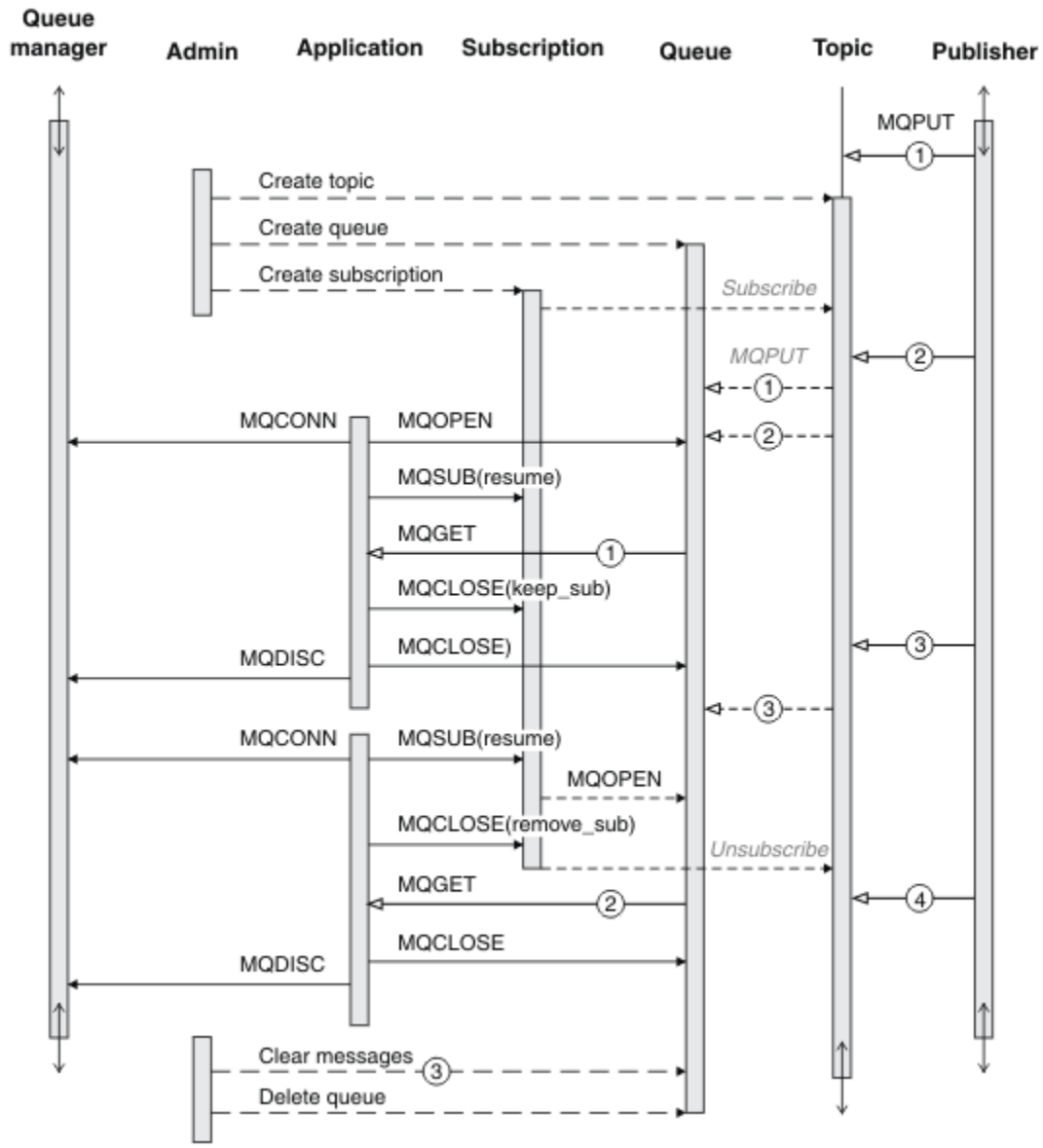
Administrátor je přidán do třetího příkladu: nespravovaný trvalý odběratel. Je to dobrý příklad, který ukazuje, jak může administrátor komunikovat s aplikací publikování/odběru.

Zobrazí se body, které se mají zobrazit.

1. Vydavatel umístí zprávu 1na téma, které později bude přidruženo k objektu tématu, který se používá pro odběr. Objekt tématu definuje řetězec tématu, který se shoduje s tématem, které bylo publikováno pomocí zástupných znaků.
2. Téma má zachované publikování.
3. Administrátor vytvoří objekt tématu, frontu a odběr. Objekt tématu a fronta musí být definovány před odběrem.
4. Aplikace otevře frontu přidruženou k odběru a předá `MQSUB` manipulátoru fronty. Může to alternativně otevřít odběr a předat mu popisovač fronty `MQHO_NONE`. Příkaz converse není pravdivý, nelze obnovit odběr tím, že mu projde pouze popisovač fronty bez názvu odběru-fronta může mít více odběrů.
5. Aplikace otevře odběr s použitím volby `MQSO_RESUME` , i když je prvním otevření odběru poprvé. Obnovuje se administrativně vytvořený odběr.
6. Odběratel obdrží zachované publikování 1. Publikování 2, ačkoli bylo publikováno před přijetím jakýchkoli publikování odběratelem, bylo publikováno po spuštění odběru a je druhé publikování ve frontě odběru.

Poznámka: Pokud zachované publikování není publikováno jako trvalá zpráva, je po restartování správce front ztraceno.

7. V tomto příkladu je odběr trvalý. Je možné, aby program vytvořil nespravovaný netrvalý odběr. Mělo by být zřejmé, že to není něco, co by měl administrátor dělat.
8. Efekt volby MQCO_REMOVE_SUB při zavření odběru je odebrání odběru, jako kdyby jej administrátor odstranil. Tím se zastaví jakékoli další publikace odesílané do fronty, ale neovlivní publikování, která jsou již ve frontě, i když je fronta uzavřena, na rozdíl od *spravovaného* trvalého odběru.
9. Administrátor později odstraní zbývající zprávu, 3a odstraní frontu.



Obrázek 95. Nespravované linie lifeline

Normální vzor pro nespravovaný odběr je určen pro úklid front a odběrů, který má provádět administrátor. Obvykle se nikdo nepokusí emulovat chování spravovaného odběratele a uklidit fronty a odběry programově v kódu aplikace. Pokud zjistíte, že potřebujete zapsat logiku správy, otázka, zda můžete dosáhnout stejných výsledků pomocí spravovaného vzorku. Není jednoduché napsat úzce synchronizovaný, zcela spolehlivý kód managementu. Je jednodušší uklidit později, buď ručně, nebo pomocí automatizovaného programu správy, když si můžete být jisti, že zprávy, odběry a fronty mohou být jednoduše vymazány, bez ohledu na jejich stav.

Vlastnosti zprávy publikování/odběru

Systém zpráv typu IBM MQ pro publikování/odběr se vztahuje k několika vlastnostem zprávy.

Token PubAccounting

Jedná se o hodnotu, která bude uvedena v poli AccountingToken deskriptoru zpráv (MQMD) všech publikovaných zpráv, které odpovídají tomuto odběru. AccountingToken je součástí kontextu identity zprávy. Další informace o kontextu zprávy viz “kontext zprávy” na stránce 41. Další informace o poli AccountingToken v produktu MQMD najdete v tématu AccountingToken.

PubApplIdentityData

Jedná se o hodnotu, která bude v poli Data ApplIdentitydeskriptoru zpráv (MQMD) všech publikovaných zpráv, odpovídajících tomuto odběru. ApplIdentityData jsou součástí kontextu identity zprávy. Další informace o kontextu zprávy viz “kontext zprávy” na stránce 41. Další informace o datovém poli ApplIdentityv produktu MQMD najdete v tématu Data aplikaceApplIdentity.

Není-li určena volba MQSO_SET_IDENTITY_CONTEXT, budou data ApplIdentity, která budou nastavena v každé zprávě publikované pro tento odběr, prázdná, jako výchozí kontextové informace.

Je-li určena volba MQSO_SET_IDENTITY_CONTEXT, generuje se uživatel PubApplIdentityData a toto pole je vstupní pole, které obsahuje data produktu ApplIdentity, která mají být nastavena v každé publikaci pro tento odběr.

PubPriority

Jedná se o hodnotu, která bude v poli Priorita deskriptoru zpráv (MQMD) všech publikovaných zpráv, odpovídajících tomuto odběru. Další informace o poli Priorita v MQMD naleznete v tématu Priorita.

Hodnota musí být větší než nula nebo rovna nule; nula je nejnižší priorita. Mohou být použity také následující speciální hodnoty:

- MQPRI_PRIORITY_AS_Q_DEF-Je-li fronta odběru uvedena v poli Hobj v rámci volání MQSUB a nejedná se o spravovanou obslužnou rutinu, bude priorita zprávy převzata z atributu DefProirity této fronty. Je-li takto označená fronta fronta klastru nebo existuje více než jedna definice v cestě rozpoznání názvu fronty, je priorita určena při vložení zprávy publikování do fronty, jak je popsáno pro položku Priorita v deskriptoru MQMD. Pokud volání MQSUB používá spravovanou obslužnou rutinu, je priorita zprávy převzata z atributu DefPriority fronty modelu přidružené k odběru tématu přihlášeným k odběru.
- MQPRI_PRIORITY_AS_PUBLISHED-Priorita pro zprávu je priorita původní publikace. Toto je počáteční hodnota tohoto pole.

SubCorrelId

Upozornění: Identifikátor korelace může být předáván pouze mezi správci front v klastru publikování/odběru, ne v hierarchii.

Všechny publikace odeslané tak, aby odpovídaly tomuto odběru, budou obsahovat tento korelační identifikátor v deskriptoru zpráv. Pokud více odběrů používá stejnou frontu k získání svých publikací, pomocí funkce MQGET podle ID korelace lze získat pouze publikování pro specifický odběr, který má být získán. Tento korelační identifikátor může vygenerovat buď správce front, nebo uživatel.

Není-li určena volba MQSO_SET_CORREL_ID, je identifikátor korelace generován správcem front a toto pole je výstupní pole, které obsahuje identifikátor korelace, který bude nastaven v každé zprávě publikované pro tento odběr.

Je-li určena volba MQSO_SET_CORREL_ID, je identifikátor korelace generován uživatelem a toto pole je vstupní pole, které obsahuje identifikátor korelace, který má být nastaven v každé publikaci pro tento odběr. V tomto případě, pokud pole obsahuje MQCI_NONE, bude korelační identifikátor, který bude nastaven v každé zprávě publikované pro tento odběr, představovat korelační identifikátor vytvořený původním vložením této zprávy.

Je-li zadána volba MQSO_GROUP_SUB a zadaný identifikátor korelace je shodný s existujícím seskupeným odběrem s použitím stejné fronty a překrývajícím se řetězcem tématu, je k dispozici pouze nejvýznamnější odběr ve skupině s kopií této publikace.

SubUserData

Jedná se o uživatelská data odběru. Data poskytnutá na odběru v tomto poli budou zahrnuta jako vlastnost datové zprávy MQSubUserpro každou publikaci odeslanou do tohoto odběru.

Vlastnosti publikování

Tabulka 110 na stránce 804 obsahuje seznam vlastností publikování, které jsou k dispozici spolu se zprávou o publikování.

K těmto vlastnostem můžete přistupovat přímo ze složky **MQRFH2** , nebo je načíst pomocí produktu MQINQMP. MQINQMP přijímá buď název vlastnosti, nebo **MQRFH2** název jako název vlastnosti, na kterou se má dotaz dotázat.

Tabulka 110. Vlastnosti publikování

Název vlastnosti	Název MQRFH2	Typ	Popis
MQTopicString	mqps.Top	MQTYPE_STRING	Řetězec tématu
MQSubUserData	mqps.Sud	MQTYPE_STRING	Uživatelská data odběratele
MQIsRetained	mqps.Ret	MQTYPE_BOOLEAN	Zachované publikování
MQPubOptions	mqps.Pub	MQTYPE_INT32	Volby publikování
MQPubLevel	mqps.Pbl	MQTYPE_INT32	Úroveň zveřejnění
MQPubTime	mqpse.Pts	MQTYPE_STRING	Čas publikování
MQPubSeqNum	mqpse.Seq	MQTYPE_INT32	Pořadové číslo publikace
MQPubStrIntData	mqpse.Sid	MQTYPE_STRING	String/Integer data přidaná vydavatelem
MQPubFormat	mqpse.Pfmt	MQTYPE_INT32	Formát zprávy: MQRFH1 MQRFH2 PCF

Uspořádání zpráv

U konkrétního tématu jsou zprávy publikovány správcem front ve stejném pořadí, v jakém jsou přijímány z publikování aplikací (je-li změna pořadí založena na prioritě zpráv).

Řazení zpráv obvykle znamená, že každý odběratel přijímá zprávy od konkrétního správce front v konkrétním tématu od určitého vydavatele v pořadí, ve kterém jsou publikovány vydavatelem.

Nicméně stejně jako u všech zpráv produktu IBM MQ je možné zprávy občas doručovat mimo pořadí. K tomu může dojít v následujících situacích:

- Je-li odkaz v síti vypnutý a následné zprávy jsou přesměrovány podél jiného odkazu
- Pokud je fronta dočasně zaplněna nebo zablokována tak, aby byla zpráva vložena do fronty nedoručených zpráv, a proto zpožděna, zatímco následné zprávy procházejí přímo.
- Pokud administrátor odstraní správce front, když jsou vydavatelé a odběratelé stále v provozu, způsobí, že zprávy ve frontě budou umístěny do fronty nedoručených zpráv a odběry mají být přerušeny.

Pokud tyto okolnosti nemohou nastat, jsou publikace vždy doručovány v uvedeném pořadí.

Poznámka: Seskupené nebo segmentované zprávy nelze použít s publikováním/odběrem.

Zachycení publikací

Můžete zachytit publikování, upravit jej a pak ji znovu publikovat, než dosáhne dalšího odběratele.

Možná budete chtít před tím, než dojde k odběrateli, zachytit publikování, aby bylo možné provést jednu z následujících akcí:

- Připojit další informace ke zprávě
- Blokovat zprávu
- Transformovat zprávu

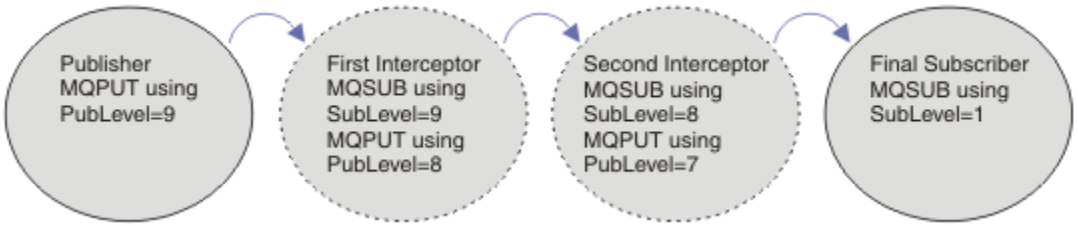
Na každé zprávě můžete provést stejnou operaci nebo můžete operaci změnit, a to v závislosti na odběru, zprávě nebo záhlaví zprávy.

Související informace

MQ_PUBLISH_EXIT-Uživatelská procedura publikování

Úrovně odběru

Nastavte úroveň odběru odběru, aby bylo možné publikování zachytit, než dosáhne svých konečných odběratelů. Zachytávající odběratel odebírá odběr na vyšší úrovni odběru a publikuje se na nižší úrovni publikování. Sestavte řetězec zachycující odběratele, aby před doručením koncovým odběratelům zpracovali zpracování zpráv na publikování.



Obrázek 96. Posloupnost zachytávajících odběratelů

Chcete-li zachytit publikování, použijte atribut **MQSD** `SubLevel` . Po zachycení zprávy lze tuto zprávu transformovat a poté znovu publikovat na nižší úrovni publikování změnou atributu **MQPMO** `PubLevel` . Zpráva pak přejde ke koncovým odběratelům, nebo je opět zadržena zprostředkujícím odběratelem na nižší úrovni odběru.

Odběratel zachycení obvykle transformuje zprávu před opětovným publikováním. Sled zpráv tvoří posloupnost zachycujících odběratelů. Eventuálně nelze znovu publikovat zachycenou publikaci: Odběratelé na nižších úrovních odběru by zprávu neobdrželi.

Ujistěte se, že zachytávač přijímá publikování před všemi ostatními odběrateli. Nastavte úroveň odběru zachytávače vyšší než ostatní odběratelé. Odběratelé standardně mají `SubLevel` produktu 1. Nejvyšší hodnota je 9. Publikování musí začínat znakem `PubLevel` alespoň stejně, jako nejvyšší `SubLevel`. Publikovat nejprve s výchozí hodnotou `PubLevel` produktu 9.

- Máte-li jedno zachycení odběratele na téma, nastavte parametr `SubLevel` na hodnotu 9.
- Pro více zakročujících aplikací na téma nastavte nižší úroveň `SubLevel` pro každého následného zachytávače zachycení.
- Můžete implementovat maximum 8 zachytávajících aplikací, s úrovněmi odběrů od 9 do 2 včetně. Konečný příjemce zprávy má `SubLevel` produktu 1.

Zachytávač s nejvyšší úrovní odběru, která se rovná nebo je nižší než `PubLevel` publikování, obdrží publikování jako první. Konfigurovat pouze jeden zachytávač odběratele pro téma na konkrétní úrovni odběru. Existence více odběratelů na určité úrovni odběru vede k více kopiím publikace zasílané do konečného souboru odebírajících aplikací.

Odběratel s hodnotou SubLevel produktu 0 se používá jako catchall. Pokud žádný koncový odběratel nedostane zprávu, obdrží tuto příručku. Odběratel s SubLevel produktu 0 může být použit k monitorování publikací, které neobdrželi jiní odběratelé.

Programování zachytávače zachycení

Použijte volby odběru popsané v tématu Tabulka 111 na stránce 806.

Tabulka 111. Volby odběru pro zachytávající odběratele	
Volba odběru	**Notes**
MQSO_SET_CORREL_ID a SubCorrelId jsou nastaveny na MQCI_NONE	Ponechte CorrelId zachycené publikace stejné jako původní publikování. **Poznámka:** Nelze předat identifikátor korelace publikování v hierarchii. Pole je používáno správcem front.
PubPriority je nastavena na MQPRI_PRIORITY_AS_PUBLISHED	Ponechejte prioritu zachycené publikace stejně jako v původní publikaci.

Volby v produktu Tabulka 111 na stránce 806 musí být použity všemi zachytávajícími odběrateli. Výsledkem je, že identifikátor korelace a priorita zprávy nejsou upraveny z nastavení původního vydavatele.

Když zachytávající odběratel zpracoval publikaci, znovu publikuje zprávu do stejného tématu na úrovni PubLevel o jednu nižší, než je SubLevel vlastního odběru. Pokud zachycovací odběratel nastavil SubLevel produktu 9, publikuje zprávu znovu s parametrem PubLevel produktu 8.

Chcete-li zprávu znovu publikovat správně, je třeba použít několik informací z původní příručky. Znovu použijte stejné **MQMD** jako v původní zprávě a nastavte MQPMO_PASS_ALL_CONTEXT , abyste se ujistili, že všechny informace v **MQMD** jsou předány dalšímu odběrateli. Zkopírujte hodnoty z vlastností zprávy, které jsou zobrazeny v Tabulka 112 na stránce 806 , do odpovídajících polí znovu publikované zprávy. Odběratel zachycení může tyto hodnoty změnit. K přidání dalších hodnot do **MQPMO**použijte operátor OR. Pole Volby slouží ke sloučení vložených voleb zpráv.

Frontu publikování musíte explicitně otevřít raději, než použít spravovanou frontu publikování. Pro spravovanou frontu nelze nastavit MQSO_SET_CORREL_ID . Také nelze nastavit MQOO_SAVE_ALL_CONTEXT ve spravované frontě. Viz fragmenty kódu uvedené v části "Příklady" na stránce 807.

Tabulka 112. Hodnoty MQPUT pro znovu publikované zprávy	
Znovu publikovat zprávu pomocí MQPUT	**Informace ve zprávě o publikování**
MQOD. ObjectString	vlastnost zprávy MQTopicString
MQPMO. Options	vlastnost zprávy MQPubOptions

Konečný odběratel má volbu nastavení voleb odběru rozdílně. Například, může nastavit prioritu publikování explicitně spíše než na MQPRI_PRIORITY_AS_PUBLISHED. Nastavení konečného odběratele ovlivní pouze publikování z konečného zachytávače odběratele v řetězci.

Zachovaná publikování

Zachované publikování musí být po zachycení uchováno zkopírováním původních voleb vložení zpráv do znovu publikované zprávy.

Volba MQPMO_RETAIN je nastavena vydavatelem. Každý zachytávací odběratel musí přenést MQPubOptions do voleb vložení zprávy znovu publikovaná zpráva, jak je zobrazeno v Tabulka 112 na

stránce 806. Kopírování voleb vkládání zpráv zachovává volby nastavené původním vydavatelem, včetně informací o tom, zda má být publikace zachována.

Jakmile publikace dokončí svůj průchod řetězu zachycení odběratelů a doručí se konečným odběratelům, je nakonec uchován. Noví odběratelé, na `SubLevel 1`, požadující zachované publikování, jej přijmou bez dalšího zaceňování. Odběratelé na `SubLevel` větší než 1 neodešlou zachované publikování. V důsledku toho zůstává zachované publikování neupravováno řetězcem zachycení odběratele podruhé zaokrouhlením.

Příklady

Příklady jsou fragmenty kódu, které lze zkombinovat k sestavení syndikátu odběratele. Kodex je napsán jako stručný, spíše než v kvalitě výroby.

Direktivy preprocesoru v produktu Obrázek 97 na stránce 807 definují dvě vlastnosti, které mají být extrahovány ze zpráv publikování, které jsou vyžadovány voláním MQI produktu `MQINQMP` .

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <cmqc.h>
#define      MQPUBOPTIONS        (MQPTR)(char*) "MQPubOptions",\
                                 0,\
                                 12,\
                                 MQVS_NULL_TERMINATED,\
                                 MQCCSI_APPL
#define      MQTOPICSTRING       (MQPTR)(char*) "MQTopicString",\
                                 0,\
                                 13,\
                                 MQVS_NULL_TERMINATED,\
                                 MQCCSI_APPL
```

Obrázek 97. Direktivy preprocesoru

Příkaz Obrázek 98 na stránce 808 vypíše deklarace použité v fragmentech kódu. S výjimkou zvýrazněných výrazů jsou deklarace standardní pro aplikaci produktu IBM MQ .

Zvýrazněné volby Put a Get jsou inicializovány tak, aby prošly celý kontext. Zvýrazněné moduly `MQTOPICSTRING` a `MQPUBOPTIONS` jsou inicializátory MQCHARV pro názvy vlastností, které jsou definovány v direktivách preprocesoru. Názvy jsou předány produktu `MQINQMP`.

```
int main(int argc, char **argv) {
    MQLONG   Reason            = MQRC_NONE;
    MQLONG   CompCode          = MQCC_OK;
    MQHCONN  Hcon              = MQHC_UNUSABLE_HCONN;
    MQCHAR   QMName[49]        = " ";
    MQCMHO   CrtMsgHOpts       = {MQCMHO_DEFAULT};
    MQHMSG   Hmsg              = MQHM_NONE;
    MQMD     md                = {MQMD_DEFAULT};
    MQHOBJ   gHobj             = MQHO_NONE;
    MQOD     getOD             = {MQOD_DEFAULT};
    MQGMO    gmo               = {MQGMO_DEFAULT};
    MQLONG   GO_Options        = MQOO_INPUT_AS_Q_DEF
                               | MQOO_FAIL_IF_QUIESCING
                               | MQOO_SAVE_ALL_CONTEXT;
    MQLONG   GC_Options        = MQCO_DELETE_PURGE;
    MQHOBJ   Hsub              = MQHO_NONE;
    MQSD     sd                = {MQSD_DEFAULT};
    MQLONG   SC_Options        = MQCO_NONE;
    MQHOBJ   pHobj             = MQHO_NONE;
    MQOD     putOD             = {MQOD_DEFAULT};
    MQLONG   PO_Options        = MQOO_OUTPUT
                               | MQOO_FAIL_IF_QUIESCING
                               | MQOO_PASS_ALL_CONTEXT;
    MQLONG   PC_Options        = MQCO_NONE;
    MQPMO    pmo               = {MQPMO_DEFAULT};
    MQIMPO   InqPropOpts       = {MQIMPO_DEFAULT};
    MQPD     PropDesc          = {MQPD_DEFAULT};
    MQLONG   Type              = MQTYPE_AS_SET;
    MQCHARV  TopStrProp        = {MQTOPICSTRING};
    MQCHARV  PubOptProp        = {MQPUBOPTIONS};
    MQLONG   DataLength        = 0;
    MQBYTE   buffer[256]       = "";
    MQLONG   buflen            = sizeof(buffer) - 1;
    MQLONG   messlen           = 0;
    char     TopStrBuf[256]    = "Initial value";
    int      i                 = 0;
```

Obrázek 98. Deklarace

Inicializace, které nejsou snadno provedeny v deklaracích, jsou zobrazeny v Obrázek 99 na stránce 809. Zvýrazněné hodnoty vyžadují vysvětlení.

SYSTEM.NDURABLE.MODEL.QUEUE
V tomto příkladu místo použití produktu MQSUB pro otevření spravovaného netrvalého odběru se modelová fronta SYSTEM.NDURABLE.MODEL.QUEUEpoužívá k vytvoření dočasné dynamické fronty. Jeho popisovač se předá do MQSUB. Otevřením fronty přímo budete schopni uložit celý kontext zprávy a nastavit volbu odběru MQSO_SET_CORREL_ID.

MQGMO_CURRENT_VERSION
Je důležité použít aktuální verzi většiny struktur produktu IBM MQ . Pole jako gmo.MsgHandle jsou k dispozici pouze v nejnovější verzi řídicích struktur.

MQGMO_PROPERTIES_IN_HANDLE
Řetězec tématu a volby vložení zpráv nastavené v původní publikaci mají být načteny zachycujícím odběratelem pomocí vlastností zprávy. Alternativou by bylo čtení struktury **MQRFH2** přímo ve zprávě.

MQSO_SET_CORREL_ID
Použijte MQSO_SET_CORREL_ID v kombinaci s,

```
memcpy(sd.SubCorrelId, MQCI_NONE, sizeof(sd.SubCorrelId));
```

Efekt těchto voleb je předání identifikátoru korelace. Identifikátor korelace nastavený původním vydavatelem je umístěn v poli identifikátoru korelace v publikování, které přijímá zachytávačovatel zachycení. Každý zachytávající odběratel přechází na stejný identifikátor korelace. Konečný odběratel pak má volbu přijetí stejného identifikátoru korelace.

Poznámka: Je-li publikace předávána prostřednictvím hierarchie publikování/odběru, nebude identifikátor korelace nikdy zachován.

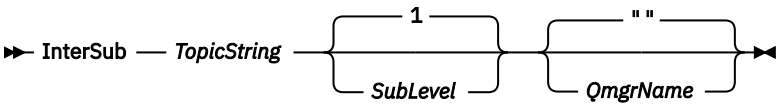
MQPRI_PRIORITY_AS_PUBLISHED
Publikace se umístí do fronty publikování se stejnou prioritou zprávy, jako byla publikována.

```
strncpy(getOD.ObjectName,    "SYSTEM.NDURABLE.MODEL.QUEUE",
        sizeof(getOD.ObjectName));
gmo.Version                = MQGMO_VERSION_4;
gmo.Options                = MQGMO_WAIT
                           | MQGMO_PROPERTIES_IN_HANDLE
                           | MQGMO_CONVERT;
gmo.WaitInterval           = 30000;
sd.Options                 = MQSO_CREATE
                           | MQSO_FAIL_IF_QUIESCING
                           | MQSO_SET_CORREL_ID;
sd.PubPriority             = MQPRI_PRIORITY_AS_PUBLISHED;
sd.Version                 = MQSD_VERSION_1;
memcpy(sd.SubCorrelId, MQCI_NONE, sizeof(sd.SubCorrelId));
putOD.ObjectType             = MQOT_TOPIC;
putOD.ObjectString.VSPtr     = &TopStrBuf;
putOD.ObjectString.VSBufSize = sizeof(TopStrBuf);
putOD.ObjectString.VSLength  = MQVS_NULL_TERMINATED;
putOD.ObjectString.VSCCSID   = MQCCSI_APPL;
putOD.Version                = MQOD_VERSION_4;
pmo.Version                = MQPMO_VERSION_3;
```

Obrázek 99. Inicializace

Obrázek 100 na stránce 810 ukazuje fragment kódu pro čtení parametrů příkazového řádku, dokončení inicializace a vytvoření odběru odběru.

Spusťte program s příkazem,



Aby zpracování chyb bylo možné bez omezení, je kód příčiny každého volání MQI uložen v odlišném prvku pole. Po každém volání kódu dokončení je testován kód dokončení a pokud je hodnota MQCC_FAIL, ovládací prvek ukončí blok kódu do `{ } while(0)`.

Dva zajímavé řádky kódu jsou,

pmo.PubLevel = sd.SubLevel - 1;
Nastaví úroveň publikování pro znovu publikovanou zprávu na nižší než úroveň odběru zachycujícího odběratele.

gmo.MsgHandle = Hmsg;
Poskytuje popisovač zprávy pro MQGET , aby vrátil vlastnosti zprávy.

```
do {
   printf("Intercepting subscriber start\n");
   if (argc < 2) {
       printf("Required parameter missing - topic string\n");
       exit(99);
   } else {
       sd.ObjectString.VSPtr    = argv[1];
       sd.ObjectString.VSLength = MQVS_NULL_TERMINATED;
       printf("TopicString = %s\n", sd.ObjectString.VSPtr);
   }
   if (argc > 2) {
       sd.SubLevel  = atoi(argv[2]);
       pmo.PubLevel = sd.SubLevel - 1;
       printf("SubLevel is %d, PubLevel is %d\n", sd.SubLevel, pmo.PubLevel);
   }
   if (argc > 3)
       strncpy(QMName, argv[3], sizeof(QMName));
   MQCONN(QMName, &Hcon, &CompCode, &Reason);
   if (CompCode == MQCC_FAILED)
       break;
   MQOPEN(Hcon, &getOD, GO_Options, &gHobj, &CompCode, &Reason);
   if (CompCode == MQCC_FAILED)
       break;
   MQSUB(Hcon, &sd, &gHobj, &Hsub, &CompCode, &Reason);
   if (CompCode == MQCC_FAILED)
       break;
   MQCRTMH(Hcon, &CrtMsgHOpts, &Hmsg, &CompCode, &Reason);
   if (CompCode == MQCC_FAILED)
       break;
   gmo.MsgHandle = Hmsg;
```

Obrázek 100. Příprava na zachycení publikování

Hlavní fragment kódu, Obrázek 101 na stránce 811, získává zprávy z fronty publikování. Dotazuje se na vlastnosti zprávy a znovu publikuje zprávy pomocí řetězce tématu a původní **MQPMO**. Vlastnosti `volby` publikace.

V tomto příkladu není na publikování provedena žádná transformace. Řetězec tématu publikované publikace se vždy shoduje s řetězcem tématu, k jehož odběru přihlášeno odběratele. Je-li zachytávající odběratel zodpovědný za zachycení více odběrů odeslaných do stejné publikační fronty, může být nutné dotaz na řetězec tématu odlišit od publikací, které odpovídají různým odběrům.

Volání na `MQINQMP` jsou zvýrazněna. Vlastnosti řetězce tématu a volby vložení vlastností zprávy jsou zapsány přímo do řídicích struktur výstupu. Jediný důvod změny pole s délkou MQCHARV souboru `putOD.ObjectString` z explicitní délky na řetězec s ukončenou hodnotou null má za účelem výstupu řetězce použít příkaz `printf` .

```
while (CompCode != MQCC_FAILED) {
   memcpy(md.MsgId, MQMI_NONE, sizeof(md.MsgId));
   memcpy(md.CorrelId, MQCI_NONE, sizeof(md.CorrelId));
   md.Encoding       = MQENC_NATIVE;
   md.CodedCharSetId = MQCCSI_Q_MGR;
   printf("MQGET : %d seconds wait time\n", gmo.WaitInterval/1000);
   MQGET(Hcon, gHobj, &md, &gmo, buflen, buffer, &messlen,
       &CompCode, &Reason);
   if (CompCode == MQCC_FAILED)
       break;
   buffer[messlen] = '\0';
   MQINQMP(Hcon, Hmsg,    &InqPropOpts, &TopStrProp, &PropDesc, &Type,
       putOD.ObjectString.VSBufSize, putOD.ObjectString.VSPtr,
       &(putOD.ObjectString.VSLength), &CompCode, &Reason);
   if (CompCode == MQCC_FAILED)
       break;
   memset((void *)((MQLONG)(putOD.ObjectString.VSPtr)
       + putOD.ObjectString.VSLength),'\0',1);
   putOD.ObjectString.VSLength = MQVS_NULL_TERMINATED;
   MQINQMP(Hcon, Hmsg, &InqPropOpts, &PubOptProp, &PropDesc, &Type,
       sizeof(pmo.Options), &(pmo.Options), &DataLength,
       &CompCode, &Reason);
   if (CompCode == MQCC_FAILED)
       break;
   MQOPEN(Hcon, &putOD, PO_Options, &pHobj, &CompCode, &Reason);
   if (CompCode == MQCC_FAILED)
       break;
   printf("Republish message <%s> on topic <%s> with options %d\n",
       buffer, putOD.ObjectString.VSPtr, pmo.Options);
   MQPUT(Hcon, pHobj, &md, &pmo, messlen, buffer, &CompCode, &Reason);
   if (CompCode == MQCC_FAILED)
       break;
   MQCLOSE(Hcon, &pHobj, PC_Options, &CompCode, &Reason);
   if (CompCode == MQCC_FAILED)
   break;
}
```

Obrázek 101. Publikování a opětovné publikování produktu Intercept

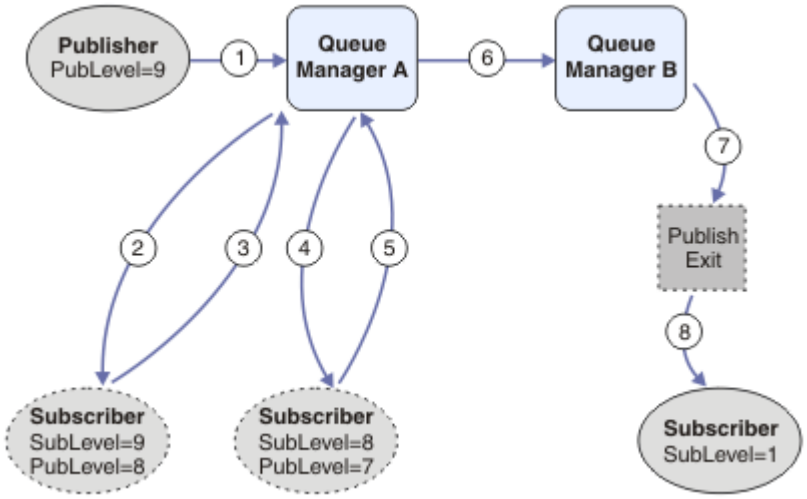
Poslední fragment kódu je zobrazen v Obrázek 102 na stránce 811.

```
    } while (0);
    if (CompCode == MQCC_FAILED && Reason != MQRC_NO_MSG_AVAILABLE)
        printf("MQI Call failed with reason code %d\n", Reason);
    if (Hsub!= MQHO_NONE)
        MQCLOSE(Hcon, &Hsub, SC_Options, &CompCode, &Reason);
    if (Hcon!= MQHC_UNUSABLE_HCONN)
        MQDISC(&Hcon, &CompCode, &Reason);
}
```

Obrázek 102. Dokončení

Zachycení publikací a distribuovaného publikování/odběru
Při implementaci zachycení odběratelů nebo publikování v rámci distribuované topologie publikování/odběru postupujte podle jednoduchého vzoru. Implementace zachytává odběratele ve stejných správcích front jako vydavatelé a Publish opouští ve stejných správcích front jako koneční odběratelé.

Produkt Obrázek 103 na stránce 812 zobrazuje dva správce front připojené v klastru odběru publikování. Vydavatel vytvoří publikování na téma klastru na úrovni publikování 9. Očíslované šipky ukazují posloupnost kroků, které byly v publikování provedeny, když se přenášejí na odběratele v tématu klastru. Publikování je zachyceno odběratelem s hodnotou `Dílčí úroveň` 9 a znovu publikovány s úrovní `Publevel` 8. Odběratel je znovu zachycen na `Sublevel` 8. Odběratel znovu publikuje na `Publevel` 7. Odběratel proxy poskytnutý správcem front postoupí publikování do správce front B, kde byla kromě konečného odběratele implementována uživatelská procedura publikování. Publikování je zpracováno uživatelskou procedurou publikování před tím, než je nakonec přijata koncovým odběratelem na `dílčí úrovni` 1. Zachycovací odběratelé a uživatelská procedura publikování se zobrazí se zalomenými obrysy.



Obrázek 103. Výslech a uživatelská procedura publikování v klastru

Cílem jednoduchého vzoru je pro každého odběratele, který přijímá publikování, aby obdržel stejnou publikaci. Publikování probíhá prostřednictvím stejné posloupnosti transformací bez ohledu na to, kde je odběratel připojen. Pravděpodobně se chcete vyhnout posloupnosti transformací, a to v závislosti na tom, kde jsou vydavatelé nebo koneční odběratelé připojeni. Přípravou výjimkou by bylo přizpůsobení příručky konečnému uživateli každému jednotlivému odběrateli. Pomocí uživatelské procedury publikování lze přizpůsobit publikování na základě fronty, do níž je publikace konečně doručena.

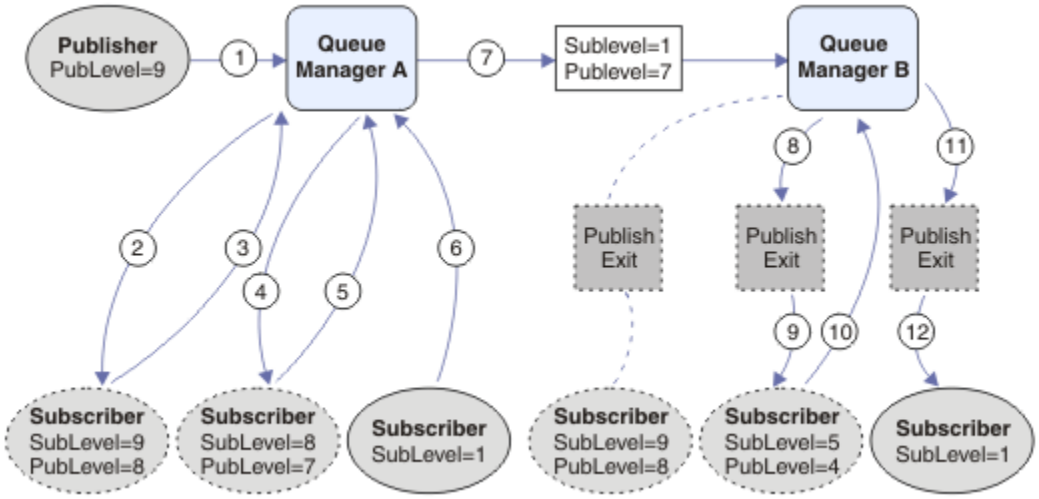
Musíte pečlivě zvážit, kam implementovat zachycení odběratele a ukončení publikování v topologii distribuovaného publikování/odběru. Přímočarý návrhový vzor implementuje zachycení odběratelů do stejného správce front jako vydavatelé a Publish exits ke stejným správcům front jako koneční odběratelé.

Anti-pattern

Obrázek 104 na stránce 813 ukazuje, jak se mohou věci navrátit, pokud nepostupujete podle jednoduchého vzoru. Chcete-li zkomplikovat implementaci, je do správce front A přidán konečný odběratel a do správce front B jsou přidáni dva další zachytávaní odběratelé.

Publikování bude předáno správci front B na úrovni `PubLevel` 7, kde je zachycen odběratelem na `SubLevel` 5 před spotřebním konečným odběratelem na `SubLevel` 1. Uživatelská procedura publikování zachycuje publikování před tím, než je předána do zachytávačů spotřebitele i u konečného spotřebitele ve správci front B. Publikování dosáhne konečného odběratele ve správci front A, aniž by bylo zpracováno uživatelskou procedurou publikování.

V topologii publikování/odběru se odběratelé proxy přihlásí k odběru na `SubLevel` 1a předávají na `PubLevel` nastaveni posledním zachycujícím odběratelem. V produktu Obrázek 104 na stránce 813je výsledkem skutečnost, že publikování není zachyceno odběratelem s použitím produktu `SubLevel` 9 ve správci front B.



Obrázek 104. Komplexní implementace zachycujících odběratelů

Volby publikování

K dispozici je několik možností, které řídí způsob, jakým jsou zprávy publikovány.

Srážková odpověď-na informace od odběratelů

Pokud nechcete, aby odběratelé mohli odpovědět na publikování, která přijímají, je možné zatajit informace v polích ReplyToQ a ReplyToQmgr deskriptoru MQMD pomocí volby put-message příkazu MQPMO_SUPPRESS_REPLYTO. Je-li tato volba použita, odebere správce front tyto informace z MQMD, jakmile obdrží publikování, než ji předá všem odběratelům.

Tuto volbu nelze použít v kombinaci s volbou sestavy, která potřebuje odpověď ReplyToQ, pokud se tento pokus o volání nezdaří s chybou MQRC_MISSING_REPLY_TO_Q.

Úroveň zveřejnění

Použití úrovní publikování je způsob, jak řídit, kteří odběratelé obdrží publikování. Úroveň publikování určuje úroveň odběru, na kterou se má publikace zaměřit. Publikaci obdrží pouze odběry s nejvyšší úrovní odběru, která je nižší než úroveň publikování nebo její úrovně zveřejnění. Tato hodnota musí být v rozsahu nula až devět; nula je nejnižší úroveň publikování. Počáteční hodnota tohoto pole je 9. Jednou z použití úrovní publikování a odběru je zachycení publikování.

Kontrola, zda není publikování doručeno žádnému odběratelům

Chcete-li zkontrolovat, zda publikování nebylo doručeno žádnému odběrateli, použijte volbu vložení zprávy MQPMO_WARN_IF_NO_SUBS_MATCHED s voláním MQPUT. Pokud operace vložení vrátí kód dokončení MQCC_WARNING a kód příčiny MQRC_NO_SUBS_MATCHED, publikování nebylo doručeno do žádných odběrů. Je-li v operaci vložení zadána volba MQPMO_RETAIN, bude zpráva uchována a doručena do všech následně definovaných odpovídajících odběrů. V distribuovaném systému publikování/odběru je návratový kód MQRC_NO_SUBS_MATCHED vrácen pouze v případě, že nejsou registrovány žádné odběry proxy pro dané téma ve správci front.

Volby odběru

K dispozici je několik možností, které řídí způsob zpracování odběrů zpráv.

Trvalost zpráv

Správci front udržují perzistenci publikací, které předávají odběratelům, jak je nastaveno vydavatelem. Vydavatel nastaví perzistenci na jednu z následujících možností:

0
Přechodné

1
Trvalý

2
Perzistence jako fronta/definice tématu

U publikování/odběru vydavatel vyřeší objekt tématu a produkt **topicString** na vyřešený objekt tématu. Pokud vydavatel určuje definici trvání jako frontu/téma, bude pro publikování nastavena výchozí perzistence z vyřešeného objektu tématu.

Zachovaná publikování

Pro řízení při přijetí zachovaných publikování mohou odběratelé používat dvě volby odběru:

Publikovat na žádost pouze, MQSO_PUBLICATIONS_ON_REQUEST

Chcete-li, aby odběratel měl kontrolu nad tím, kdy přijímá publikace, můžete použít volbu odběru MQSO_PUBLICATIONS_ON_REQUEST. Odběratel může poté řídit, kdy obdrží publikování, pomocí volání MQSUBRQ (zadání manipulátoru Hsub, který byl vrácen z původního volání MQSUB) k odeslání požadavku na jeho zachované publikování. Odběratelé používající volbu odběru MQSO_PUBLICATIONS_ON_REQUEST nepřijímají žádné nezachované publikace.

Uvedete-li MQSO_PUBLICATIONS_ON_REQUEST, musíte použít MQSUBRQ k načtení libovolné publikace. Pokud nepoužíváte funkci MQSO_PUBLICATIONS_ON_REQUEST, získáte zprávy při jejich publikování.

Pokud odběratel používá volání MQSUBRQ a používá zástupné znaky v tématu odběru, může odběr odpovídat více tématům nebo uzlům ve stromu témat, přičemž všechny zachované zprávy (pokud nějaké existují) budou odeslány odběrateli.

Tato volba může být užitečná zejména při použití s trvalým odběrem, protože správce front bude i nadále odesílat publikace odběrateli, pokud je trvale odebírán, a to i v případě, že aplikace odběratele není spuštěna. To by mohlo vést k hromadným zprávám ve frontě odběratele. Tomuto sestavení lze se vyhnout, pokud se odběratel registruje pomocí volby MQSO_PUBLICATIONS_ON_REQUEST. Případně můžete použít netrvalé odběry, pokud je to vhodné pro vaši aplikaci, abyste se vyvarovali vytváření nežádoucích zpráv.

Je-li odběr trvalý a vydavatel používá zachované publikace, může aplikace odběratele použít volání MQSUBRQ k aktualizaci informací o stavu po restartu. Odběratel musí poté pravidelně aktualizovat svůj stav pomocí volání MQSUBRQ.

Pomocí této volby nebudou odeslány žádné publikace jako výsledek volání MQSUB. Trvalý odběr, který byl obnoven po odpojení, bude používat volbu MQSO_PUBLICATIONS_ON_REQUEST, pokud byl pro použití této volby konfigurován původní odběr.

Pouze nové publikování, pouze MQSO_NEW_PUBLICATIONS_ONLY

Pokud v tématu existuje zachované publikování, obdrží všechny odběratele, kteří provádějí odběr po publikování, kopii této publikace. Pokud odběratel nechce přijímat žádná publikování, která byla provedena dříve, než je předplatný odběr, může odběratel použít volbu odběru MQSO_NEW_PUBLISATIONS_ONLY.

Seskupení odběrů

Uvažte seskupení odběrů, pokud jste nastavili frontu pro příjem publikování a máte-li počet překrývajících se odběrů, které dodávají do stejné fronty. Tato situace je podobná jako příklad v tématu Překrývání odběrů.

Když se přihlásíte k odběru tématu, můžete se vyhnout příjmu duplicitních publikování nastavením volby MQSO_GROUP_SUB . Výsledkem je, že pokud se více než jeden odběr ve skupině shoduje s tématem publikování, odpovídá za umístění publikování do fronty pouze jeden odběr. Další odběry, které se shodují s tématem publikování, jsou ignorovány.

Odběr, který je odpovědný za umístění publikování do fronty, je vybrán na základě toho, že má nejdelší odpovídající řetězec tématu, než se narazí na libovolné zástupné znaky. Může být považováno za nejbližší odpovídající odběr. Jeho vlastnosti jsou šířeny do publikování, včetně toho, zda má vlastnost MQSO_NOT_OWN_PBS . Pokud ano, nebude do fronty doručena žádná publikace, i když jiná odpovídající předplatná nemusí mít vlastnost MQSO_NOT_OWN_PUBS .

Nemůžete umístit všechny své odběry do jedné skupiny, abyste eliminovali duplicitní publikace. Seskupené odběry musí splňovat tyto podmínky:

1. Žádná z odběrů není spravována.
2. Skupina odběrů doručují publikace do stejné fronty.
3. Každý odběr musí být na stejné úrovni odběru.
4. Zpráva o publikování pro každý odběr ve skupině má stejný korelační identifikátor.

   Chcete-li zajistit, aby každý odběr měl ve zprávě publikování se stejným identifikátorem korelace, nastavte parametr MQSO_SET_CORREL_ID pro vytvoření vlastního identifikátoru korelace v rámci publikování a nastavte stejnou hodnotu v poli **SubCorrelId** v každém odběru. Nenastavujte **SubCorrelId** na hodnotu MQCI_NONE.

Inquaning about a nastavení atributů objektu

Atributy jsou vlastnosti, které definují charakteristiky objektu IBM MQ .

Dotýkají se způsobu, jakým správce front zpracovává objekt. Atributy každého typu objektu IBM MQ jsou podrobně popsány v části Atributy objektů.

Některé atributy jsou nastaveny při definování objektu a lze je měnit pouze pomocí příkazů IBM MQ ; příklad takového atributu je výchozí prioritou pro zprávy zařazené do fronty. Ostatní atributy jsou ovlivněny operací správce front a mohou se v průběhu času měnit; příkladem je aktuální hloubka fronty.

Můžete se dotázat na aktuální hodnoty většiny atributů pomocí volání MQINQ. Modul MQI také poskytuje volání MQSET, pomocí kterého můžete změnit některé atributy fronty. Volání MQI nelze použít ke změně atributů žádného jiného typu objektu. Místo toho musíte použít jeden z následujících prostředků:

- ULW Prostředek MQSC, který je popsán v tématu Odkaz na MQSC.
- IBM i CL příkazy CHGMQMx, které jsou popsány v tématu Odkaz na příkazy jazyka CL pro produkt IBM inebo v zařízení MQSC.
- z/OS Příkazy operátoru ALTER nebo příkazy DEFINE s volbou REPLACE, které jsou popsány v příkazech MQSC.

Poznámka: Názvy atributů objektů jsou zobrazeny v této dokumentaci v podobě, kterou používáte s voláními MQINQ a MQSET. Když použijete příkazy IBM MQ k definování, změně nebo zobrazení atributů, musíte identifikovat atributy pomocí klíčových slov zobrazených v popisech příkazů v odkazech témat.

Volání MQINQ i volání MQSET používají pole selektorů k identifikaci ty atributy, které chcete dotázat nebo nastavit. Pro každý atribut, se kterým můžete pracovat, je selektor. Název selektoru má předponu určenou charakterem atributu:

Tabulka 113. Předpony názvů selektorů

Předpona	**Popis**
MQCA_	Tyto selektory odkazují na atributy, které obsahují znaková data (například název fronty).
MQIA_	Tyto selektory odkazují na atributy, které obsahují buď číselné hodnoty (jako například *CurrentQueueDepth*, počet zpráv ve frontě) nebo konstantní hodnotu (jako například *SyncPoint*, zda správce front podporuje synchronizační body).

Před použitím volání MQINQ nebo MQSET musí být vaše aplikace připojena ke správci front a vy musíte použít volání MQOPEN k otevření objektu pro nastavení nebo zjišťování informací o atributech. Tyto operace jsou popsány v "Připojování k správci front a odpojování od něj" na stránce 704 a "Otevírání a zavírání objektů" na stránce 712.

Následující odkazy vám pomohou zjistit další informace o získávání dotazu a nastavení atributů objektu:

- "Inquaktování o atributech objektu" na stránce 816
- "Některé případy, kdy volání MQINQ selže" na stránce 817
- "Nastavení atributů fronty" na stránce 817

Související pojmy

"Přehled rozhraní fronty zpráv" na stránce 690
Zde jsou uvedeny informace o komponentách rozhraní MQI (Message Queue Interface).

"Připojování k správci front a odpojování od něj" na stránce 704
Chcete-li používat programovací služby IBM MQ , musí mít program připojení ke správci front. V této části se dozvíte, jak se připojit k správci front a odpojit je od správce front.

"Otevírání a zavírání objektů" na stránce 712
Tyto informace poskytují vhled do otevírání a zavírání objektů IBM MQ .

"Vložení zpráv do fronty" na stránce 723
V této části se dozvíte, jak vložit zprávy do fronty.

"Získávání zpráv z fronty" na stránce 737
Tyto informace použijte k získání informací o získávání zpráv z fronty.

"Potvrzení a zálohování jednotek práce" na stránce 818
Tyto informace popisují, jak potvrdit a zazálohovat všechny zotavitelné operace get a put, které se vyskytly v jednotce práce.

"Spuštění aplikací produktu IBM MQ pomocí spouštěčů" na stránce 829
Informace o spouštěčích a o tom, jak spustit aplikace IBM MQ pomocí spouštěčů.

"Práce s MQI a klastry" na stránce 847
Existují speciální volby na voláních a návratových kódech, které se vztahují ke klastrování.

"Použití a zápis aplikací v systému IBM MQ for z/OS" na stránce 851
Aplikace produktu IBM MQ for z/OS mohou být vytvořeny z programů spuštěných v mnoha různých prostředích. To znamená, že mohou využít výhody zařízení dostupných ve více než jednom prostředí.

"Aplikace mostu IMS a IMS v systému IBM MQ for z/OS" na stránce 57
Tyto informace vám pomohou při psaní aplikací produktu IMS pomocí produktu IBM MQ.

Inquaktování o atributech objektu

Použijte volání MQINQ k dotazům na atributy libovolného typu IBM MQ.

Jako vstup pro toto volání musíte dodat:

- Popisovač připojení.
- Popisovač objektu.
- Počet selektorů.
- Pole selektorů atributů, každý selektor má tvar MQCA_ * nebo MQIA_ *. Každý selektor představuje atribut s hodnotou, o které se chcete dotázat, a každý selektor musí být platný pro typ objektu, který popisovač objektu představuje. Selektory můžete určit v libovolném pořadí.
- Počet celočíselných atributů, o které se budete dotazovat. Uveďte nulu, pokud se nedotazujete na celočíselné atributy.
- Délka vyrovnávací paměti znakových atributů v *CharAttrLength*. Musí to být alespoň součet délek požadovaných k zadržení každého znakového řetězce atributu. Uveďte nulu, pokud se nedotazujete na znakové atributy.

Výstup z MQINQ je:

- Sada celočíselných hodnot atributů kopírovaných do pole. Počet hodnot je určen produktem *IntAttrCount*. Pokud je hodnota *IntAttrCount* nebo *SelectorCount* nula, tento parametr se nepoužije.
- Vyrovnávací paměť, ve které jsou vráceny znakové atributy. Délka vyrovnávací paměti je dána parametrem **CharAttrLength** . Pokud je hodnota *CharAttrLength* nebo *SelectorCount* nula, tento parametr se nepoužije.
- Kód dokončení. Pokud kód dokončení vydá varování, znamená to, že volání bylo dokončeno pouze částečně. V takovém případě zkontrolujte kód příčiny.
- Kód příčiny. K dispozici jsou tři situace s částečnými dokončením:
  - Selektor není použit pro daný typ fronty.
  - Pro celočíselné atributy není k dispozici dostatek místa.
  - Pro atributy znaků není k dispozici dostatek místa.

  Pokud nastane více než jedna z těchto situací, bude vrácena první z těchto situací.

Pokud otevřete frontu pro výstup nebo dotaz a přeloží se na nemístnou frontu klastru, můžete se dotázat pouze na název fronty, typ fronty a společné atributy. Hodnoty obecných atributů jsou hodnoty zvolené fronty, pokud byla použita hodnota MQOO_BIND_ON_OPEN. Hodnoty jsou hodnoty libovolné z možných front klastru, pokud bylo použito buď MQOO_BIND_NOT_FIXED nebo MQOO_BIND_ON_GROUP, nebo MQOO_BIND_AS_Q_DEF, a atribut fronty **DefBind** byl MQBND_BIND_NOT_FIXED. Další informace viz “MQOPEN a klastry” na stránce 848 a MQOPEN .

Poznámka: Hodnoty vrácené voláním jsou snímkem vybraných atributů. Tyto atributy se mohou změnit dříve, než se váš program bude chovat na vrácených hodnotách.

V adresáři MQINQje uveden popis volání MQINQ.

Některé případy, kdy volání MQINQ selže

Pokud otevřete alias k dotazům na jeho atributy, vrátíte se do atributů alias fronty (objekt IBM MQ používaný pro přístup k jiné frontě), nikoli z základní fronty.

Avšak definice základní fronty, na kterou se alias řeší, je také otevřena správcem front, a pokud jiný program změní použití základní fronty v intervalu mezi voláními MQOPEN a MQINQ, volání MQINQ selže a vrátí kód příčiny MQRC_OBJECT_CHANGED. Volání také selže, jestliže se změní atributy objektu alias fronty.

Podobně platí, že když otevřete vzdálenou frontu a dotáže se na její atributy, vrátíte se pouze na atributy lokální definice vzdálené fronty.

Uvedete-li jeden nebo více selektorů, které nejsou platné pro typ atributů fronty, na které jste se dotazovali, volání MQINQ se dokončí s varováním a nastaví výstup následujícím způsobem:

- Pro celočíselné atributy jsou nastaveny odpovídající prvky proměnné *IntAttrs* na hodnotu MQIAV_NOT_APPLICABLE.
- Pro znakové atributy jsou odpovídající části řetězce *CharAttrs* nastaveny na hvězdičky.

Uvedete-li jeden nebo více selektorů, které nejsou platné pro typ atributů objektu, na které jste se dotazovali, volání MQINQ selže a vrátí kód příčiny MQRC_SELECTOR_ERROR.

Nelze volat MQINQ k zobrazení modelové fronty; lze použít buď prostředek MQSC, nebo příkazy dostupné na vaší platformě.

Nastavení atributů fronty

V této části se dozvíte, jak nastavit atributy fronty pomocí volání MQSET.

Pomocí volání MQSET můžete nastavit pouze následující atributy fronty:

- *InhibitGet* (ale ne pro vzdálené fronty)
- *DistList* (není na z/OS)
- *InhibitPut*

- *TriggerControl*
- *TriggerType*
- *TriggerDepth*
- *TriggerMsgPriority*
- *TriggerData*

Volání MQSET má stejné parametry jako volání MQINQ. Pro funkci MQSET jsou však všechny parametry s výjimkou kódu dokončení a kódu příčiny vstupní parametry. K dispozici nejsou žádné situace s částečnými dokončením.

Poznámka: Rozhraní MQI nelze použít k nastavení atributů jiných objektů produktu IBM MQ než lokálně definovaných front.

Další informace o volání MQSET naleznete v části MQSET.

Potvrzení a zálohování jednotek práce

Tyto informace popisují, jak potvrdit a zazálohovat všechny zotavitelné operace get a put, které se vyskytly v jednotce práce.

V tomto tématu jsou použity následující termíny:

- Potvrdit
- Vrátit zpět
- Koordinace synchronizačních bodů
- Synchronizační bod
- Pracovní jednotka
- jednofázové potvrzení
- Dvoufázové potvrzení

Pokud jste obeznámeni s těmito podmínkami zpracování transakcí, můžete přejít k tématu “Aspekty synchronizačních bodů v aplikacích produktu IBM MQ” na stránce 819.

Potvrdit a odvrátit

Když program vloží zprávu do fronty v rámci pracovní jednotky, tato zpráva se zviditelní ostatním programům pouze v případě, že program potvrdí jednotku práce. Chcete-li potvrdit jednotku práce, musí být všechny aktualizace úspěšné, aby se zachovala integrita dat. Pokud program zjistí chybu a rozhodne se, že operace vložení není trvalá, může vrátit jednotku práce zpět. Když program provádí odvolání, produkt IBM MQ obnoví frontu odebráním zpráv, které byly vloženy do fronty touto jednotkou práce. Způsob, jakým program provádí operace commit a back out, závisí na prostředí, ve kterém je program spuštěn.

Podobně platí, že když program dostane zprávu z fronty v rámci pracovní jednotky, zůstává tato zpráva ve frontě, dokud program nepotvrdí jednotku práce, ale zpráva není k dispozici pro načtení jinými programy. Zpráva je trvale odstraněna z fronty, když program potvrdí jednotku práce. Pokud program zálohuje pracovní jednotku, produkt IBM MQ obnoví frontu tím, že zpřístupní zprávy, které mají být načteny jinými programy.

Koordinace synchronizačních bodů, synchronizační bod, jednotka práce

Koordinace synchronizačních bodů je proces, při kterém jsou jednotky práce buď potvrzeny, nebo zálohovány s integritou dat.

Rozhodnutí o provedení změn nebo odvolání změn se provádí v nejjednodušším případě na konci transakce. Může však být užitečnější, aby aplikace synchronizovala změny dat na jiných logických bodech v rámci transakce. Tyto logické body se nazývají *synchronizační body* (nebo *synchronizační body*). a období zpracování sady aktualizací mezi dvěma synchronizacím se nazývá *jednotka práce*. Několik volání MQGET a volání MQPUT může být součástí jediné jednotky práce.

Maximální počet zpráv v rámci jednotky práce může být řízen atributem MAXUMSGS příkazu ALTER QMGR .

jednofázové potvrzení
Proces *jednofázové potvrzení* je takový, ve kterém může program potvrdit aktualizace fronty bez koordinace změn s ostatními správci prostředků.

Dvoufázové potvrzení
Proces *Dvoufázové potvrzení* je takový, v němž jsou aktualizace, které program provedl ve frontách produktu IBM MQ , koordinovány s aktualizacemi jiných prostředků (například databáze pod kontrolou produktu Db2). Pod takovým procesem jsou aktualizace všech prostředků potvrzeny nebo zálohovány společně.

Pro práci s jednotkami práce poskytuje produkt IBM MQ atribut **BackoutCount** . Tato hodnota se zvýší pokaždé, když se zazálohuje zpráva v rámci pracovní jednotky. Pokud zpráva opakovaně způsobí, že jednotka práce abnormálně skončí, hodnota *BackoutCount* nakonec překročí hodnotu zadanou *BackoutThreshold*. Tato hodnota je nastavena, když je definována fronta. V takové situaci může aplikace odebrat zprávu z pracovní jednotky a vložit ji do jiné fronty, jak je definováno v produktu *BackoutRequeueQName*. Když je zpráva přesunuta, může jednotka práce potvrdit.

Následující odkazy použijte k vyhledání dalších informací o potvrzení a zálohování jednotek práce:

- “Aspekty synchronizačních bodů v aplikacích produktu IBM MQ” na stránce 819
- z/OS “Synchronizační body v aplikacích produktu IBM MQ for z/OS” na stránce 821
- IBM i “Synchronizační body v produktu CICS pro aplikace produktu IBM i” na stránce 823
- “Synchronizační body v produktu IBM MQ for Multiplatforms” na stránce 823
- IBM i “Rozhraní pro externí správce synchronizačního bodu produktu IBM i” na stránce 828

Související pojmy
“Přehled rozhraní fronty zpráv” na stránce 690
Zde jsou uvedeny informace o komponentách rozhraní MQI (Message Queue Interface).

“Připojování k správci front a odpojování od něj” na stránce 704
Chcete-li používat programovací služby IBM MQ , musí mít program připojení ke správci front. V této části se dozvíte, jak se připojit k správci front a odpojit je od správce front.

“Otevírání a zavírání objektů” na stránce 712
Tyto informace poskytují vhled do otevírání a zavírání objektů IBM MQ .

“Vložení zpráv do fronty” na stránce 723
V této části se dozvíte, jak vložit zprávy do fronty.

“Získávání zpráv z fronty” na stránce 737
Tyto informace použijte k získání informací o získávání zpráv z fronty.

“Inquaning about a nastavení atributů objektu” na stránce 815
Atributy jsou vlastnosti, které definují charakteristiky objektu IBM MQ .

“Spuštění aplikací produktu IBM MQ pomocí spouštěčů” na stránce 829
Informace o spouštěčích a o tom, jak spustit aplikace IBM MQ pomocí spouštěčů.

“Práce s MQI a klastry” na stránce 847
Existují speciální volby na voláních a návratových kódech, které se vztahují ke klastrování.

“Použití a zápis aplikací v systému IBM MQ for z/OS” na stránce 851
Aplikace produktu IBM MQ for z/OS mohou být vytvořeny z programů spuštěných v mnoha různých prostředích. To znamená, že mohou využít výhody zařízení dostupných ve více než jednom prostředí.

“Aplikace mostu IMS a IMS v systému IBM MQ for z/OS” na stránce 57
Tyto informace vám pomohou při psaní aplikací produktu IMS pomocí produktu IBM MQ.

Aspekty synchronizačních bodů v aplikacích produktu IBM MQ
Tyto informace použijte k seznámení se s použitím synchronizačních bodů v aplikacích produktu IBM MQ .

Dvoufázové potvrzování je podporováno v následujících prostředích:

- AIX IBM MQ for AIX
- IBM i IBM MQ for IBM i
- HP-UX IBM MQ for HP-UX
- Linux IBM MQ for Linux
- Solaris IBM MQ for Solaris
- Windows IBM MQ for Windows
- z/OS CICS Transaction Server pro z/OS
- z/OS TXSeries
- z/OS IMS/ESA
- z/OS Dávka produktu z/OS s RRS
- Další externí koordinátoři pomocí rozhraní X/Open XA

Jednofázové potvrzování je podporováno v následujících prostředích:

- IBM i IBM MQ for IBM i
- UNIX IBM MQ na platformě UNIX
- Windows IBM MQ for Windows
- z/OS z/OS dávkové

Další informace o externích rozhraních viz “Rozhraní pro externí správce synchronizačního bodu na platformách Multiplatforms” na stránce 826a dokumentaci XA *CAE Specification Distributed Transaction Processing: The XA Specification*, publikované skupinou Open Group. Správci transakcí (například CICS, IMS, Encina a Tuxedo) se mohou podílet na dvoufázovém potvrzování koordinovaném s jinými zotavitelné prostředky. To znamená, že funkce vytváření front poskytované produktem IBM MQ mohou být přeneseny do rozsahu pracovní jednotky, kterou spravuje správce transakcí.

Ukázky dodávané s produktem IBM MQ zobrazují IBM MQ koordinující databáze vyhovující standardu XA. Další informace o těchto ukázkách naleznete v tématu “Použití ukázkových procedurálních programů produktu IBM MQ” na stránce 1034.

Ve své aplikaci IBM MQ můžete v každé operaci vkládání a volání určit, zda má být volání pod řízením synchronizačního bodu. Chcete-li provést operaci vložení pod řízením synchronizačního bodu, použijte hodnotu MQPMO_SYNCPOINT v poli *Options* struktury MQPMO, když voláte příkaz MQPUT. V případě operace get použijte hodnotu MQGMO_SYNCPOINT v poli *Options* struktury MQGMO. Pokud volbu nevyberete explicitně, závisí výchozí akce na platformě:

- Multi Výchozí ovládací prvek synchronizačního bodu je NO.
- z/OS Výchozí ovládací prvek synchronizačního bodu je YES.

Je-li volání MQPUT1 vydáno s MQPMO_SYNCPOINT, změní se výchozí chování, takže je operace vložení dokončena asynchronně. To může způsobit změnu chování některých aplikací, které závisí na určitých polích ve strukturách MQOD a MQMD, které jsou vraceny, ale které nyní obsahují nedefinované hodnoty. Aplikace může určit MQPMO_SYNC_RESPONSE, aby se zajistilo, že je operace umístění provedena synchronně a že jsou dokončeny všechny odpovídající hodnoty polí.

Když aplikace obdrží kód příčiny MQRC_BACLED_OUT v odpovědi na synchronizační bod MQPUT nebo MQGET v rámci synchronizačního bodu, měla by aplikace za normálních okolností vrátit aktuální transakci pomocí operace MQBACK a poté v případě potřeby zkusit znovu celou transakci. Pokud aplikace přijme odpověď MQRC_BACKED_OUT v odpovědi na volání MQCMIT nebo MQDISC, není nutné volat MQBACK.

Při každém vrácení volání MQGET je zvýšena hodnota pole *BackoutCount* struktury MQMD ovlivněné zprávy. Vysoká hodnota *BackoutCount* označuje zprávu, která byla opakovaně vrácena. To může naznačovat problém s touto zprávou, kterou byste měli vyšetřit. Podrobnosti viz BackoutCount , kde najdete podrobnosti o *BackoutCount*.

S výjimkou z/OS dávky s RRS, pokud program vydá volání MQDISC, když existují nepotvrzené požadavky, dojde k implicitnímu synchronizačního bodu. Pokud se program ukončí nestandardně, dojde k implicitnímu vrácení.

z/OS V systému z/OSdojde k implicitnímu synchronizačním bodu také v případě, že program skončí normálně bez prvního volání funkce MQDISC. Má se za to, že program skončil normálně, pokud je TCB připojená k produktu MQ normálně ukončena. Při spuštění v produktu z/OS UNIX System Services and Language Environment (LE) je vyvolání výchozí podmínky vyvoláno za účelem abend nebo signálů. Obslužné rutiny podmínky LE zpracují chybový stav a TCB se normálně ukončí. Za těchto podmínek produkt MQ potvrdí jednotku práce. Další informace naleznete v tématu Úvod do zpracování podmínek jazykového prostředí.

z/OS Pro programy IBM MQ for z/OS můžete použít volbu MQGMO_MARK_SKIP_BACKU_BOUT k určení, že zpráva nesmí být vrácena, dojde-li k odvolání (aby se zabránilo smyčky *MQGET-error-backout*). Informace o použití této volby najdete v tématu “Vynechání odvolání” na stránce 766.

Změny atributů fronty (volání MQSET nebo pomocí příkazů) nejsou ovlivněny potvrzením nebo zálohováním jednotek práce.

z/OS *Synchronizační body v aplikacích produktu IBM MQ for z/OS*

Toto téma vysvětluje, jak používat synchronizační body ve správci transakcí (CICS a IMS) a dávkové aplikace.

z/OS *Synchronizační body v produktu CICS Transaction Server pro aplikace produktu z/OS*

V aplikaci CICS zavedete synchronizační bod pomocí příkazu EXEC CICS SYNCPOINT.

Chcete-li provést vrácení všech změn předchozího synchronizačního bodu, můžete použít příkaz EXEC CICS SYNCPOINT ROLLBACK. Další informace naleznete v příručce *CICS Application Programming Reference*.

Pokud se do jednotky práce podílejí jiné obnovitelné prostředky, správce front (ve spojení se správcem synchronizačního bodu CICS) se podílí na protokolu dvoufázového potvrzování. V opačném případě správce front provede proces jednofázového potvrzení.

Pokud aplikace CICS vydá volání MQDISC, nedojde k žádnému implicitnímu synchronizačního bodu. Pokud se aplikace zavře normálně, všechny otevřené fronty se zavřou a dojde k implicitnímu odevzdání. Pokud se aplikace ukončí nestandardně, všechny otevřené fronty se zavřou a dojde k implicitnímu vrácení.

z/OS *Synchronizační body v aplikacích produktu IMS*

V aplikaci IMS vytvořte synchronizační bod pomocí volání příkazu IMS , jako je GU (get unique) na IOPCB a CHKP (kontrolní bod).

Chcete-li odvolat všechny změny od předchozího kontrolního bodu, můžete použít volání IMS ROLB (rollback). Další informace naleznete v dokumentaci produktu IMS .

Správce front (ve spojení se správcem synchronizačního bodu IMS) se podílí na protokolu s dvoufázovým potvrzováním, pokud se do jednotky práce podílejí i jiné obnovitelné prostředky.

Všechny otevřené manipulátory jsou uzavřeny adaptérem IMS v synchronizačním bodu (s výjimkou dávkového nebo neřízeného prostředí BMP). Důvodem je skutečnost, že jiný uživatel by mohl zahájit další jednotku práce a kontrola zabezpečení produktu IBM MQ se provádí při volání MQCONN, MQCONNX a MQOPEN, nikoli při volání MQPUT nebo MQGET.

V prostředí WFI (Wait-for-Input) nebo PWFI (pseudo Wait-for-Input) IMS však IBM MQ neuzavřete obslužné rutiny, dokud nebude doručena další zpráva nebo dokud nebude do aplikace vrácen stavový kód QC. Pokud aplikace čeká v oblasti IMS a některá z těchto obslužných rutin patří ke spouštěcím frontám,

spuštění se neproběhne, protože fronty jsou otevřené. Z tohoto důvodu by aplikace spuštěné v prostředí WFI nebo PWFI měly explicitně MQCLOSE před provedením příkazu GU na port IOPCB pro další zprávu, než provede toto zpracování.

Je-li při volání MQDISC použita aplikace IMS (buď BMP nebo MPP), otevřené fronty jsou zavřeny, ale není proveden žádný implicitní synchronizační bod. Pokud se aplikace zavře normálně, všechny otevřené fronty se zavřou a dojde k implicitnímu odevzdání. Pokud se aplikace ukončí nestandardně, všechny otevřené fronty se zavřou a dojde k implicitnímu vrácení.

z/OS *Synchronizační body v dávkových aplikacích produktu z/OS*

Pro dávkové aplikace můžete použít volání správy synchronizačních bodů produktu IBM MQ : MQCMIT a MQBACK. Kvůli kompatibilitě se staršími verzemi jsou CSQBCMT a CSQBBAK k dispozici jako synonyma.

Poznámka: Pokud potřebujete potvrzovat nebo zálohovat aktualizace prostředků spravovaných různými správci prostředků, například IBM MQ a Db2, v rámci jediné jednotky práce můžete použít službu RRS. Další informace viz “Správa transakcí a zotavitelné služby správce prostředků” na stránce 822.

Potvrzení změn pomocí volání MQCMIT

Jako vstup musíte dodat manipulátor připojení (*Hconn*), který je vrácen voláním MQCONN nebo MQCONNX.

Výstup z MQCMIT je kód dokončení a kód příčiny. Volání bude dokončeno s varováním, pokud byl synchronizační bod dokončen, ale správce front zazálohoval operace put a get od předchozího synchronizačního bodu.

Úspěšné dokončení volání MQCMIT znamená správci front, že aplikace dosáhla synchronizačního bodu a že všechny operace put a get byly provedeny od předchozího synchronizačního bodu jako trvalé.

Ne všechny odpovědi na selhání znamenají, že objekt MQCMIT nebyl dokončen. Aplikace může například přijmout MQRC_CONNECTION_BROKEN.

V adresáři MQCMITje uveden popis volání MQCMIT.

Zálohování změn pomocí volání MQBACK

Jako vstup musíte dodat popisovač připojení (*Hconn*). Použijte popisovač, který je vrácen voláním MQCONN nebo MQCONNX.

Výstup z MQBACK je kód dokončení a kód příčiny.

Výstup označuje správci front, že aplikace dosáhla synchronizačního bodu a že všechny operace get a put byly provedeny od posledního synchronizačního bodu.

K dispozici je popis volání MQBACK v MQBACK.

Správa transakcí a zotavitelné služby správce prostředků

Služby správy transakcí a zotavitelné služby správce prostředků (RRS) představují prostředek produktu z/OS k poskytování podpory dvoufázového synchronizačního bodu v rámci zúčastněných správců prostředků.

Aplikace může aktualizovat zotavitelné prostředky spravované různými správci prostředků z/OS , například IBM MQ a Db2, a poté tyto aktualizace potvrdit nebo vrátit jako jedinou jednotku práce. RRS poskytuje během normálního provedení protokolování stavu jednotky práce, koordinuje zpracování synchronizačního bodu a během restartu subsystému poskytuje odpovídající informace o stavu jednotky práce.

Podpora účastníka RRS IBM MQ for z/OS umožňuje aplikacím produktu IBM MQ v prostředí dávkových úloh, TSO a Db2 aktualizovat prostředky produktu IBM MQ i jiné prostředky nežIBM MQ (například Db2). v rámci jedné logické jednotky práce. Informace o podpoře účastníka RRS najdete v příručce *z/OS MVS Programming: Resource Recovery*.

Aplikace IBM MQ může použít buď MQCMIT a MQBACK, nebo ekvivalentní volání RRS, SRRCMIT a SRRBACK. Další informace viz “Dávkový adaptér RRS” na stránce 854.

Dostupnost RRS

Pokud není služba RRS na vašem systému z/OS aktivní, volání příkazu IBM MQ vydané z programu propojeného buď stubem RRS (CSQBRSTB nebo CSQBRRSI) vrátí MQRC_ENVIRONMENT_ERROR.

Db2Uložené procedury

Používáte-li uložené procedury produktu Db2 s produktem RRS, uvědomte si následující skutečnosti:

- Uložené procedury produktu Db2 , které používají službu RRS, musí být spravováno správcem pracovní zátěže (WLM-managed).
- Obsahuje-li uložená procedura Db2volání IBM MQ a je propojena buď s voláním stubu RRS (CSQBRSTB nebo CSQBRRSI), volání MQCONN nebo MQCONNX vrátí hodnotu MQRC_ENVIRONMENT_ERROR.
- Pokud uložená procedura spravovanou správcem WLM obsahuje volání produktu IBM MQ a je propojena s voláním jiného typu než RRS, volání MQCONN nebo MQCONNX vrátí MQRC_ENVIRONMENT_ERROR, pokud se nejedná o první volání IBM MQ provedené od spuštění adresního prostoru uložené procedury.
- Pokud uložená procedura produktu Db2 obsahuje volání produktu IBM MQ a je propojena se stubem jiného typu než RRS, prostředky produktu IBM MQ aktualizované v této uložené proceduře nebudou potvrzeny, dokud nedojde k ukončení adresního prostoru uložené procedury nebo až do následné uložené procedury MQCMIT (pomocí stubu IBM MQ Batch/TSO).
- Více kopií stejné uložené procedury může být spuštěno souběžně ve stejném adresovém prostoru. Pokud chcete, aby program Db2 použil jednu kopii uložené procedury, zkontrolujte, zda je váš program kódován způsobem znovu. Jinak byste mohli obdržet MQRC_HCONN_ERROR ve kterémkoli volání programu IBM MQ ve vašem programu.
- Nekódovat MQCMIT nebo MQBACK v uložené proceduře Db2 spravované WLM.
- Navrhněte všechny programy, které se mají spustit v prostředí Language Environment (LE).

IBM i *Synchronizační body v produktu CICS pro aplikace produktu IBM i*

IBM MQ for IBM i se podílí na CICS pro IBM i jednotky práce. Pomocí modulu MQI lze v produktu CICS pro aplikaci IBM i vkládat a získávat zprávy v rámci aktuální jednotky práce.

Příkaz EXEC CICS SYNCPOINT můžete použít k vytvoření synchronizačního bodu, který zahrnuje operace produktu IBM MQ for IBM i . Chcete-li odvolat všechny změny až na předchozí synchronizační bod, můžete použít příkaz EXEC CICS SYNCPOINT ROLLBACK.

If you use MQPUT, MQPUT1, or MQGET with the MQPMO_SYNCPOINT, or MQGMO_SYNCPOINT, option set in a CICS for IBM i application, you cannot log off CICS for IBM i until IBM MQ for IBM i has removed its registration as an API commitment resource. Před odpojením od správce front proveďte operaci commit nebo back out any pending put nebo get operations. To vám umožní odhlásit se CICS pro IBM i.

Multi *Synchronizační body v produktu IBM MQ for Multiplatforms*

Podpora synchronizačních bodů funguje na dvou typech jednotek práce: lokální a globální.

Lokální jednotka práce je taková, v níž jsou jedinými aktualizovanými prostředky správce front produktu IBM MQ . Zde je koordinace synchronizačního bodu zajišťována samotným správcem front pomocí procedury jednofázového potvrzování.

Globální jednotka práce je taková, v níž jsou aktualizovány také prostředky náležící jiným správcům prostředků, jako jsou databáze. Produkt IBM MQ může koordinovat takové jednotky práce samotné. Mohou být také koordinovány externím vázaným řadičem. Příklad:

- Jiný správce transakcí
- IBM i Řadič vázaného zpracování produktu IBM i

Pro úplnou integritu použijte proceduru s dvoufázovým potvrzováním. Dvoufázové potvrzování mohou být poskytovány správci transakcí a databázemi kompatibilními s podporou standardu XA. Příklad:

- TXSeries
- UDB
- IBM i Řadič vázaného zpracování IBM i

ULW Produkty IBM MQ mohou koordinovat globální jednotky práce pomocí dvoufázového procesu vázaného zpracování.

IBM i Produkt IBM MQ for IBM i může vystupovat jako správce prostředků pro globální jednotky práce v rámci prostředí produktu WebSphere Application Server , ale nemůže vystupovat jako správce transakcí.

Implicitní synchronizační bod

V 9.0.5

Při ukládání trvalých zpráv je produkt IBM MQ optimalizován pro ukládání trvalých zpráv v rámci synchronizačního bodu. Pokud tyto aplikace používají synchronizační bod, je lepší provádět více aplikací umísťujících trvalé zprávy do stejné fronty. Důvodem je skutečnost, že pro danou frontu je méně soupeření, pokud se synchronizační bod používá k umístění trvalých zpráv.

Produkt **ImplSyncOpenOutput** přidá implicitní synchronizační bod, když aplikace umístí trvalé zprávy mimo synchronizační bod. To poskytuje zlepšení výkonu, aniž by aplikace byly informovány o implicitním synchronizačním bodu.

Implicitní synchronizační bod poskytuje pouze zvýšení výkonu, když existuje více aplikací putování do fronty, protože snižuje soupeření o frontu. Takže **ImplSyncOpenOutput** určuje minimální počet aplikací, které mají otevřenou frontu pro výstup před přidáním implicitního synchronizačního bodu. Výchozí hodnota je 2. To znamená, že pokud nezadáte **ImplSyncOpenOutput**, implicitní synchronizační bod je přidán pouze v případě, že do fronty vkládáte více aplikací.

Další informace viz Parametry ladění .

Místní jednotky práce na více platformách
Jednotky práce, které zahrnují pouze správce front, se nazývají *lokální* jednotky práce. Koordinace synchronizačního bodu je poskytována samotným správcem front (interní koordinace) pomocí procesu jednofázového potvrzení.

Chcete-li spustit lokální jednotku práce, aplikace vydá požadavky MQGET, MQPUT nebo MQPUT1 s určením příslušné volby synchronizačního bodu. Jednotka práce je potvrzena pomocí MQCMIT nebo odvolána pomocí operace MQBACK. Pracovní jednotka je však také ukončena, když je přerušeno spojení mezi aplikací a správcem front úmyslně nebo neúmyslně.

Pokud se aplikace odpojí od správce front (MQDISC), zatímco globální transakce koordinovaná produktem IBM MQ je stále aktivní, dojde k pokusu o potvrzení jednotky práce. Je-li však aplikace ukončena bez odpojení, jednotka práce je odvolána, protože se má za to, že se žádost ukončila abnormálně.

Globální jednotky práce na více platformách
Použijte globální jednotky práce, pokud budete také muset zahrnout aktualizace prostředků náležících do jiných správců prostředků.

Zde může být koordinace interní nebo externí pro správce front:

Interní koordinace synchronizačního bodu

Koordinace globálních transakcí správce front není podporována produktem IBM MQ for IBM i nebo IBM MQ for z/OS. Nepodporuje se v prostředí IBM MQ MQI client.

Zde IBM MQ provádí koordinaci. Chcete-li spustit globální jednotku práce, aplikace vydá volání MQBEGIN.

Jako vstup do volání MQBEGIN je třeba zadat manipulátor připojení (*Hconn*) vrácený voláním MQCONN nebo MQCONNX. Tento manipulátor představuje připojení ke správci front produktu IBM MQ .

Aplikace vydá požadavky MQGET, MQPUT nebo MQPUT1 s určením příslušné volby synchronizačního bodu. To znamená, že můžete pomocí funkce MQBEGIN zahájit globální pracovní jednotku, která aktualizuje lokální prostředky, prostředky patřící jiným správcům prostředků, nebo obojí. Aktualizace prostředků náležejících k jiným správcům prostředků jsou prováděny pomocí rozhraní API daného správce prostředků. Rozhraní MQI však nelze použít k aktualizaci front, které patří k jiným správcům front. Před spuštěním dalších jednotek práce (lokální nebo globální) zadejte MQCMIT nebo MQBACK.

Globální pracovní jednotka je potvrzena pomocí MQCMIT, což iniciuje dvoufázové potvrzování všech správců prostředků zapojených do pracovní jednotky. Proces dvoufázového potvrzování se používá v případech, kdy jsou správci prostředků (například správci databází kompatibilní s XA jako Db2, Oraclea Sybase) vyzváni, aby se připravovali na potvrzení. Pouze v případě, že jsou všechny připraveny, jsou vyzvány k odevzdání. Pokud některý správce prostředků signalizuje, že jej nemůže potvrdit, bude každý z nich požádán o vrácení zpět. Případně můžete použít příkaz MQBACK k odvolání aktualizací všech správců prostředků.

Pokud se aplikace odpojí (MQDISC), zatímco je stále aktivní globální transakce, jednotka práce je potvrzena. Je-li však aplikace ukončena bez odpojení, jednotka práce je odvolána, protože se má za to, že se žádost ukončila abnormálně.

Výstup z MQBEGIN je kód dokončení a kód příčiny.

Při spuštění globální jednotky práce pomocí příkazu MQBEGIN jsou zahrnuty všechny externí správce prostředků, kteří byli konfigurováni s použitím správce front. Volání však spustí jednotku práce, ale dokončí se s varováním, pokud:

- Neexistují žádné zúčastněné správce prostředků (to znamená, že u správce front nebyli konfigurováni žádní správci prostředků)

, nebo

- Jeden nebo více správců prostředků není k dispozici.

V těchto případech musí jednotka práce zahrnovat aktualizace pouze na ty správce prostředků, kteří byli dostupní při spuštění jednotky práce.

Pokud jeden ze správců prostředků nemůže potvrdit své aktualizace, jsou správci prostředků vyzváni k odvolání jejich aktualizací a produkt MQCMIT je dokončen s varováním. Za neobvyklých okolností (obvykle zásah operátora) může volání MQCMIT selhat, pokud některé správce prostředků potvrdí své aktualizace, ale ostatní je vrátí zpět; práce je považována za dokončenou s výsledkem *smíšený* výsledek. Tyto výskyty jsou diagnostikovány v protokolu chyb správce front, aby bylo možné provést nápravnou akci.

MQCMIT globální transakce uspěje v případě, že všechny správci prostředků potvrdí své aktualizace.

Popis volání MQBEGIN viz MQBEGIN.

Koordinace externího synchronizačního bodu

K tomu dojde, pokud byl vybrán jiný koordinátor synchronizačního bodu než IBM MQ ; například CICS, Encina nebo Tuxedo.

V této situaci produkt IBM MQ v systémech UNIX and Linux a IBM MQ for Windows zaregistruje svůj zájem na výsledku transakce s koordinátorem synchronizačního bodu, aby bylo možné potvrdit nebo odvolat všechny nepotvrzené operace get nebo put, jak je požadováno. Koordinátor externího synchronizačního bodu určuje, zda jsou poskytnuty jednofázové a dvoufázové protokoly závazků.

Při použití externího koordinátora nelze vydat příkaz MQCMIT, MQBACK a MQBEGIN. Volání těchto funkcí se nezdaří s kódem příčiny MQRC_ENVIRONMENT_ERROR.

Způsob, jakým je externě koordinovaná jednotka práce spuštěna, závisí na programovacím rozhraní poskytovaném koordinátorem synchronizačního bodu. Je možné, že se požaduje explicitní volání. Je-li vyžadováno explicitní volání a zadáte-li volbu MQPMO_SYNCPOINT při spouštění jednotky práce, bude vrácen kód dokončení MQRC_SYNCPINT_NOT_AVAILABLE.

Rozsah jednotky práce je určen koordinátorem synchronizačního bodu. Stav připojení mezi aplikací a správcem front má vliv na úspěch nebo selhání volání MQI, které vyvolává problémy aplikace, nikoli stav jednotky práce. Aplikace může například odpojit a znovu připojit ke správci front během aktivní pracovní jednotky a provádět další operace MQGET a MQPUT uvnitř stejné jednotky práce. Tento stav je znám jako nevyřízené odpojení.

Volání rozhraní API produktu IBM MQ můžete použít v programech produktu CICS , ať už se rozhodnete použít možnosti XA produktu CICS. Pokud nepoužíváte volbu XA, operace vložení a získávání zpráv do front a z nich nebudou spravovány v rámci CICS atomických jednotek práce. Jedním z důvodů pro výběr této metody je to, že celková konzistence jednotky práce není pro vás důležitá.

Je-li pro vás důležitá integrita vašich jednotek práce, musíte použít standard XA. Když použijete standard XA, CICS používá protokol dvoufázového potvrzování k zajištění toho, aby všechny prostředky v rámci jednotky práce byly aktualizovány společně.

Další informace o nastavení podpory transakcí naleznete v tématu Scénáře transakčního podporya v dokumentaci produktu TXSeries CICS , například v příručce *TXSeries for Multiplatforms CICS Administration Guide for Open Systems*.

Multi V 9.0.5 *Implicitní synchronizační bod na platformách Multiplatforms*

Produkt IBM MQ 9.0.5 přidává implicitní synchronizační bod pro trvalé zprávy vnějšku synchronizačního bodu.

Při ukládání trvalých zpráv je produkt IBM MQ optimalizován pro ukládání trvalých zpráv v rámci synchronizačního bodu. Více aplikací souběžně umisťování trvalých zpráv do stejné fronty obvykle provádí lepší výkon v případě, že tyto aplikace používají synchronizační bod. Důvodem je skutečnost, že strategie zamykání produktu IBM MQ je efektivnější, pokud se při ukládání trvalých zpráv používá synchronizační bod.

Parametr **ImplSyncOpenOutput** v souboru `qm.ini` určuje, zda lze přidat implicitní synchronizační bod při vkládání trvalých zpráv aplikací mimo synchronizační bod. To může přinést zlepšení výkonu, aniž by aplikace byly informovány o implicitním synchronizačním bodu.

Implicitní synchronizační bod poskytuje zvýšení výkonu pouze v případě, že existuje více aplikací souběžně putování do fronty, protože snižuje soupeření o zámky. Hodnota **ImplSyncOpenOutput** určuje minimální počet aplikací, které mají otevřenou frontu pro výstup, než lze přidat implicitní synchronizační bod. Výchozí hodnota je *2*. To znamená, že pokud explicitně nezadáte **ImplSyncOpenOutput**, implicitní synchronizační bod je přidán pouze v případě, že do fronty vkládáte více aplikací.

Přidáte-li implicitní synchronizační bod, zobrazí se statistiky, které se projeví, a může se zobrazit výstup transakce z produktu **runmqsc display conn**.

Pokud nikdy nechcete přidat implicitní synchronizační bod, nastavte parametr **ImplSyncOpenOutput**=*OFF* .

Další informace viz Parametry ladění .

Rozhraní pro externí správce synchronizačního bodu na platformách Multiplatforms

Produkt IBM MQ for Multiplatforms podporuje koordinaci transakcí od externích správců synchronizačního bodu, kteří používají rozhraní XA sdružení X/Open XA.

Někteří správci transakcí XA (TXSeries) vyžadují, aby každý správce prostředků XA doddat svůj název. Jedná se o řetězec s názvem `name` ve struktuře přepínačů XA. Správce prostředků pro produkt IBM MQ v systému UNIX, Linux, and Windows je pojmenován MQSeries_XA_RMI. IBM i Pro produkt IBM ije název správce prostředků MQSeries XA RMI. Další podrobnosti o rozhraních XA najdete v dokumentaci XA *CAE Specification Distributed Transaction Processing: The XA Specification*(Specifikace XA) publikovaná skupinou Open Group.

V případě konfigurace XA splňuje produkt IBM MQ for Multiplatforms roli správce prostředků XA. Koordinátor synchronizačních bodů XA může spravovat sadu správců prostředků XA a synchronizovat

potvrzení nebo vrácení transakcí v obou správcích prostředků. Takto pracuje staticky zaregistrovaný správce prostředků:

1. Aplikace oznámí koordinátorovi synchronizačního bodu, že chce spustit transakci.
2. Koordinátor syncpoint vydá výzvu všem správcům prostředků, o kterých ví, že je upozorní na aktuální transakci.
3. Aplikace vydá výzvu k aktualizaci prostředků spravovaných správci prostředků přidruženým k aktuální transakci.
4. Aplikace požaduje, aby koordinátor syncpoint buď potvrdil, nebo odvolal transakci.
5. Koordinátor synchronizačního bodu vydá volání každému správci prostředků pomocí protokolů s dvoufázovým potvrzováním k dokončení transakce, jak je požadováno.

Specifikace standardu XA vyžaduje, aby každý správce prostředků poskytoval strukturu s názvem Přepínač XA. Tato struktura deklaruje schopnosti správce prostředků a funkce, které mají být volány koordinátorem synchronizačního bodu.

K dispozici jsou dvě verze této struktury:

Tabulka 114. Verze přepínače XA

Verze	**Popis**
MQRMIXASwitch	Správa statických prostředků XA
MQRMIXASwitchDynamic	Správa dynamických prostředků XA

Seznam knihoven obsahujících tuto strukturu naleznete v části Struktura přepínače IBM MQ XA.

Metoda, která musí být použita k propojení s koordinátorem synchronizačního bodu XA, je definována koordinátorem; v dokumentaci poskytnuté tímto koordinátorem zjistěte, jak povolit produktu IBM MQ spolupracovat s koordinátorem synchronizačního bodu XA.

Struktura *xa_info* , která se předává všem voláním *xa_open* koordinátorem synchronizačního bodu, může být název správce front, který má být spravován. Jedná se o stejný formulář jako název správce front předaný MQCONN nebo MQCONNX a může být prázdný, pokud má být použit výchozí správce front. Avšak můžete použít dva extra parametry TPM a AXLIB

Produkt TPM vám umožňuje uvést IBM MQ název správce transakcí, například CICS. Systém AXLIB umožňuje určit skutečný název knihovny ve správci transakcí, ve kterém jsou umístěny vstupní body XA AX.

Pokud použijete některý z těchto parametrů nebo jiného než výchozího správce front, musíte zadat název správce front pomocí parametru QMNAME. Další informace naleznete v části Parametry CHANNEL, TRPTYPE, CONNAME a QMNAME řetězce xa_open.

Omezení

1. Globální jednotky práce nejsou povoleny se sdíleným Hconn (jak je popsáno v “Sdílená připojení (nezávislá na podprocesech) s MQCONNX” na stránce 710.
2. IBM i Produkt IBM MQ for IBM i nepodporuje dynamickou registraci správců prostředků XA.

   Jediný podporovaný správce transakcí je WebSphere Application Server.
3. V systémech Windows jsou všechny funkce deklarované v přepínači XA deklarovány jako funkce _cdecl.
4. Koordinátor externího synchronizačního bodu může v daném okamžiku spravovat pouze jednoho správce front. Důvodem je skutečnost, že koordinátor má efektivní připojení ke každému správci front, a proto podléhá pravidlu, že v daném okamžiku je povoleno pouze jedno připojení.

   Poznámka: Poznámka: Aplikace klienta JMS (aplikace CLIENT JEE) spuštěná na serveru JEE nemá toto omezení, takže jedna transakce spravovaná serverem JEE může zkoordinovat více správců

front v rámci stejné transakce. Nicméně aplikace serveru JMS , která běží v režimu vazeb, je stále předmětem pravidla, že v daném okamžiku je povoleno pouze jedno připojení.

5. Všechny aplikace, které jsou spuštěny pomocí koordinátora synchronizačního bodu, se mohou připojit pouze ke správci front, který je spravován koordinátorem, protože jsou již k tomuto správci front účinně připojeni. Musí vydat příkaz MQCONN nebo MQCONNX k získání manipulátoru připojení a před ukončením operace musí vydat MQDISC. Jinou možností je použití uživatelské procedury UE014015 pro TXSeries CICS.

IBM i *Rozhraní pro externí správce synchronizačního bodu produktu IBM i*

Produkt IBM MQ for IBM i může používat nativní vázané zpracování IBM i jako externí koordinátor synchronizačního bodu.

Připojení nezávislé na vláknu (sdílená) nejsou povolena s vázaným zpracováním. Další informace o možnostech vázaného zpracování produktu IBM inajdete v publikaci *IBM i Programming: Backup and Recovery Guide, SC21-8079* .

Chcete-li spustit zařízení pro vázané zpracování IBM i , použijte systémový příkaz STRCMTCTL. K ukončení vázaného zpracování použijte příkaz systému ENDCMTCTL.

Poznámka: Výchozí hodnota *Rozsah definice vázaného zpracování* je *ACTGRP. Musí být definováno jako *JOB pro IBM MQ for IBM i. Příklad:

```
STRCMTCTL LCKLVL(*ALL) CMTSCOPE(*JOB)
```

Produkt IBM MQ for IBM i může také provádět lokální jednotky práce obsahující pouze aktualizace prostředků produktu IBM MQ . Výběr mezi lokálními jednotkami práce a účastí v globálních jednotkách práce koordinovaných produktem IBM i se provádí v každé aplikaci, když aplikace volá MQPUT, MQPUT1nebo MQGET, při specifikaci MQPMO_SYNCPOINT nebo MQGMO_SYNCPOINT nebo MQBEGIN. Pokud není vázané zpracování aktivní, když je vydáno první takové volání, produkt IBM MQ spustí lokální jednotku práce a všechny další jednotky práce pro toto připojení k produktu IBM MQ také používají lokální pracovní jednotky, bez ohledu na to, zda je poté spuštěn vázané zpracování. Chcete-li potvrdit lokální pracovní jednotku, použijte funkci MQCMIT. Chcete-li zálohovat lokální pracovní jednotku, použijte funkci MQBACK. Volání IBM i commit a rollback, jako je příkaz CL COMMIT, nemají žádný vliv na IBM MQ lokálních jednotek práce.

Chcete-li použít produkt IBM MQ for IBM i s nativním vázaným zpracováním IBM i jako externího koordinátora synchronizačního bodu, ujistěte se, že všechny úlohy s vázaným zpracováním jsou aktivní a že používáte produkt IBM MQ v úloze s jedním vláknem. Pokud zavoláte operaci MQPUT, MQPUT1nebo MQGET, při uvedení hodnoty MQPMO_SYNCPOINT nebo MQGMO_SYNCPOINT do úlohy s podporou podprocesů, ve které bylo zahájeno zpracování vázaného zpracování, volání selže s kódem příčiny MQRC_SYNCPINT_NOT_AVAILABLE.

Je možné použít lokální jednotky práce a volání MQCMIT a MQBACK ve vícevláknové úloze.

Pokud zavoláte operaci MQPUT, MQPUT1nebo MQGET, při spuštění vázaného zpracování vázaného zpracování zadejte MQPMO_SYNCPOINT nebo MQGMO_SYNCPOINT, IBM MQ for IBM i se přidá jako prostředek vázaného zpracování rozhraní API do definice vázaného zpracování. Obvykle se jedná o první takové volání v úloze. Zatímco v určité definici vázaného zpracování jsou registrovány prostředky vázaného zpracování API, nemůžete pro tuto definici ukončit vázané zpracování.

Produkt IBM MQ for IBM i při odpojování od správce front odebere svou registraci jako prostředek závazku rozhraní API, pokud v aktuální jednotce práce nejsou žádné nevyřízené operace MQI.

Pokud se odpojíte od správce front, dokud nejsou v aktuální pracovní jednotce provedeny nevyřízené operace MQPUT, MQPUT1nebo MQGET, produkt IBM MQ for IBM i zůstane registrován jako prostředek vázaného zpracování API, takže bude upozorněn na další potvrzení nebo odvolání. Když je dosaženo dalšího synchronizačního bodu, IBM MQ for IBM i potvrdí nebo odvolá změny podle potřeby. Aplikace se může odpojit a znovu připojit ke správci front během aktivní transakce a provádět další operace MQGET a MQPUT uvnitř stejné jednotky práce (jedná se o nevyřízený odpojení).

Pokusíte-li se vydat systémový příkaz ENDCMTCTL pro tuto definici vázaného zpracování, zobrazí se zpráva CPF8355 označující, že nevyřízené změny byly aktivní. Tato zpráva se také objeví v protokolu úlohy při ukončení úlohy. Chcete-li tomu zabránit, potvrďte nebo odvolat všechny nevyřízené operace IBM MQ for IBM i a odpojte se od správce front. Proto pomocí příkazů COMMIT nebo ROLLBACK dříve než ENDCMTCTL umožňuje úspěšně dokončit zpracování konce vázaného zpracování.

Při použití vázaného zpracování produktu IBM i jako externího koordinátora synchronizačního bodu nemůžete volat volání MQCMIT, MQBACK a MQBEGIN. Volání těchto funkcí se nezdaří s kódem příčiny MQRC_ENVIRONMENT_ERROR.

Chcete-li potvrdit nebo odvolat (to znamená odvolat) vaši jednotku práce, použijte jeden z programovacích jazyků, které podporují vázané zpracování. Příklad:

- Příkazy CL: COMMIT a ROLLBACK
- Programovací funkce ILE C: _Rcommit a _Rrollback
- ILE RPG: COMMIT a ROLBK
- COBOL/400: COMMIT a ROLLBACK

Při použití vázaného zpracování produktu IBM i jako externího koordinátora synchronizačních bodů s produktem IBM MQ for IBM iprovádí produkt IBM i dvoufázový potvrzovací protokol, do kterého se produkt IBM MQ podílí. Vzhledem k tomu, že každá jednotka práce je potvrzena ve dvou fázích, může být správce front nedostupný pro druhou fázi poté, co v první fázi hlasoval pro potvrzení. K tomu může dojít například v případě, že jsou ukončeny vnitřní úlohy správce front. V této situaci protokol úlohy provádějící potvrzení obsahuje zprávu CPF835F označující, že se operace commit nebo rollback nezdařila. Předchozí zprávy označují příčinu problému, ať už se vyskytla během operace commit nebo rollback, a také ID logické jednotky práce (LUWID) pro nezdařenou jednotku práce.

Pokud byl problém způsoben selháním prostředku závazků API IBM MQ během potvrzení nebo odvolání připravené transakce, můžete použít příkaz WRKMQMTRN k dokončení operace a k obnovení integrity transakce. Příkaz vyžaduje, abyste znali identifikátor LUWID pracovní jednotky pro potvrzení a vrácení zpět.

Spuštění aplikací produktu IBM MQ pomocí spouštěčů

Informace o spouštěčích a o tom, jak spustit aplikace IBM MQ pomocí spouštěčů.

Některé aplikace produktu IBM MQ , které obsluhují fronty, běží nepřetržitě, takže jsou vždy k dispozici pro načtení zpráv, které přicházejí do front. Pokud však počet příchozích zpráv doručené do front není možný, může to být nepředvídatelné. V takovém případě mohou aplikace spotřebovávat systémové prostředky i v případě, že nejsou k dispozici žádné zprávy k načtení.

IBM MQ poskytuje zařízení, které umožňuje automatické spuštění aplikace, když jsou k dispozici zprávy k načtení. Tato služba je známá jako *spouštějící*.

Informace o spouštění kanálů naleznete v části Spouštěcí kanály.

Co se spouští?

Správce front definuje určité podmínky jako *spouštěcí události*.

Je-li pro frontu povoleno spouštění a dojde k události spouštěče, správce front odešle *zprávu spouštěče* do fronty nazývané *inicializační fronta*. Přítomnost zprávy spouštěče v inicializační frontě indikuje, že došlo k události spouštěče.

Zprávy spouštěče generované správcem front nejsou trvalé. Tím se sníží protokolování (výsledkem je zlepšení výkonu) a minimalizace duplikátů během restartu, takže se zlepší doba restartování.

Program, který zpracovává inicializační frontu, se nazývá *aplikace monitoru spouštěčů*a její funkcí je číst zprávu spouštěče a provést odpovídající akci na základě informací obsažených ve zprávě spouštěče. Obvykle má tato akce spustit některou jinou aplikaci pro zpracování fronty, která generovala zprávu spouštěče. Z pohledu správce front není k dispozici žádná speciální informace o aplikaci pro monitor spouštěčů; jedná se pouze o jinou aplikaci, která čte zprávy z fronty (inicializační fronta).

Pokud je pro frontu povoleno spouštění, můžete k němu přidružit *objekt definice procesu* přidružený k této frontě. Tento objekt obsahuje informace o aplikaci, která zpracovává zprávu, která způsobila událost spouštěče. Je-li vytvořen objekt definice procesu, správce front extrahuje tyto informace a umístí jej do zprávy spouštěče, kterou použije aplikace pro monitor spouštěčů. Název definice procesu přidružený ke frontě je dán atributem local-queue produktu *ProcessName* . Každá fronta může určovat jinou definici procesu a několik front může sdílet stejnou.

Pokud chcete spustit spuštění kanálu, není nutné definovat objekt definice procesu. Místo toho se použije definice přenosové fronty.

Spouštěcí impuls je podporován klienty IBM MQ spuštěnými na serveru UNIX, Linux, and Windows. Aplikace běžící v prostředí klienta je stejná jako aplikace spuštěná v úplném prostředí produktu IBM MQ , kromě toho, že ji propojíte s knihovnami klienta. Avšak spouštěcí monitor a aplikace, která má být spuštěny, musí být ve stejném prostředí.

Spouštění zahrnuje:

Fronta aplikací
Fronta aplikací je lokální fronta, která, když má spouštěcí impuls nastavený a jsou-li splněny podmínky, vyžaduje, aby byly zapsány zprávy spouštěče.

Definice procesu
K frontě aplikací může být přidružen *objekt definice procesu* , který obsahuje podrobné informace o aplikaci, která získá zprávy z aplikační fronty. (Viz Atributy pro definice procesu pro seznam atributů.)

Nezapomeňte, že pokud chcete, aby spouštěč spouštěl kanál, není třeba definovat objekt definice procesu.

Přenosová fronta
Chcete-li spustit kanál, potřebujete přenosovou frontu, chcete-li spustit spouštěč.

Pro přenosovou frontu na jiné platformě než Linuxmůže atribut *TriggerData* přenosové fronty určovat název kanálu, který má být spuštěn. Tímto způsobem lze nahradit definici procesu pro spouštěcí kanály, ale používá se pouze v případě, že definice procesu není vytvořena.

událost spouštěče
Událost spouštěče je událost, která způsobí generování zprávy spouštěče správcem front. Obvykle se jedná o zprávu přicházející do aplikační fronty, ale může se vyskytnout i v jiných časech. Například viz “Podmínky pro událost spouštěče” na stránce 835.

IBM MQ má rozsah voleb, které vám umožňují řídit podmínky, které způsobí událost triggeru (viz “Řízení událostí spouštěče” na stránce 839).

zpráva spouštěče
Správce front vytvoří *spouštěcí zprávu* , když rozpozná událost triggeru. Zkopíruje do zprávy spouštěče informace o aplikaci, která má být spuštěna. Tyto informace pocházejí z fronty aplikace a objektu definice procesu přidruženého k aplikační frontě.

Zprávy spouštěče mají pevný formát (viz “Formát zpráv spouštěče” na stránce 846).

Inicializační fronta
Inicializační fronta je lokální fronta, do níž správce front vkládá zprávy spouštěče. Všimněte si, že inicializační fronta nemůže být alias fronta nebo modelová fronta.

Správce front může vlastnit více než jednu inicializační frontu a každý z nich je přidružen k jedné nebo více frontám aplikace.

z/OS Sdílená fronta, lokální fronta přístupná pro správce front ve skupině sdílení front, může být inicializační frontou v systému IBM MQ for z/OS.

monitor spouštěčů
Monitor spouštěčů je nepřetržitý spuštěný program, který obsluhuje jednu nebo více inicializačních front. Když do inicializační fronty přijde zpráva spouštěče, načte tuto zprávu monitor spouštěčů. Monitor spouštěčů používá informace ve zprávě spouštěče. Vydává příkaz ke spuštění aplikace, která má načíst zprávy přicházející do fronty aplikací a předávají ji informace obsažené v záhlaví zprávy spouštěče, která obsahuje název fronty aplikací.

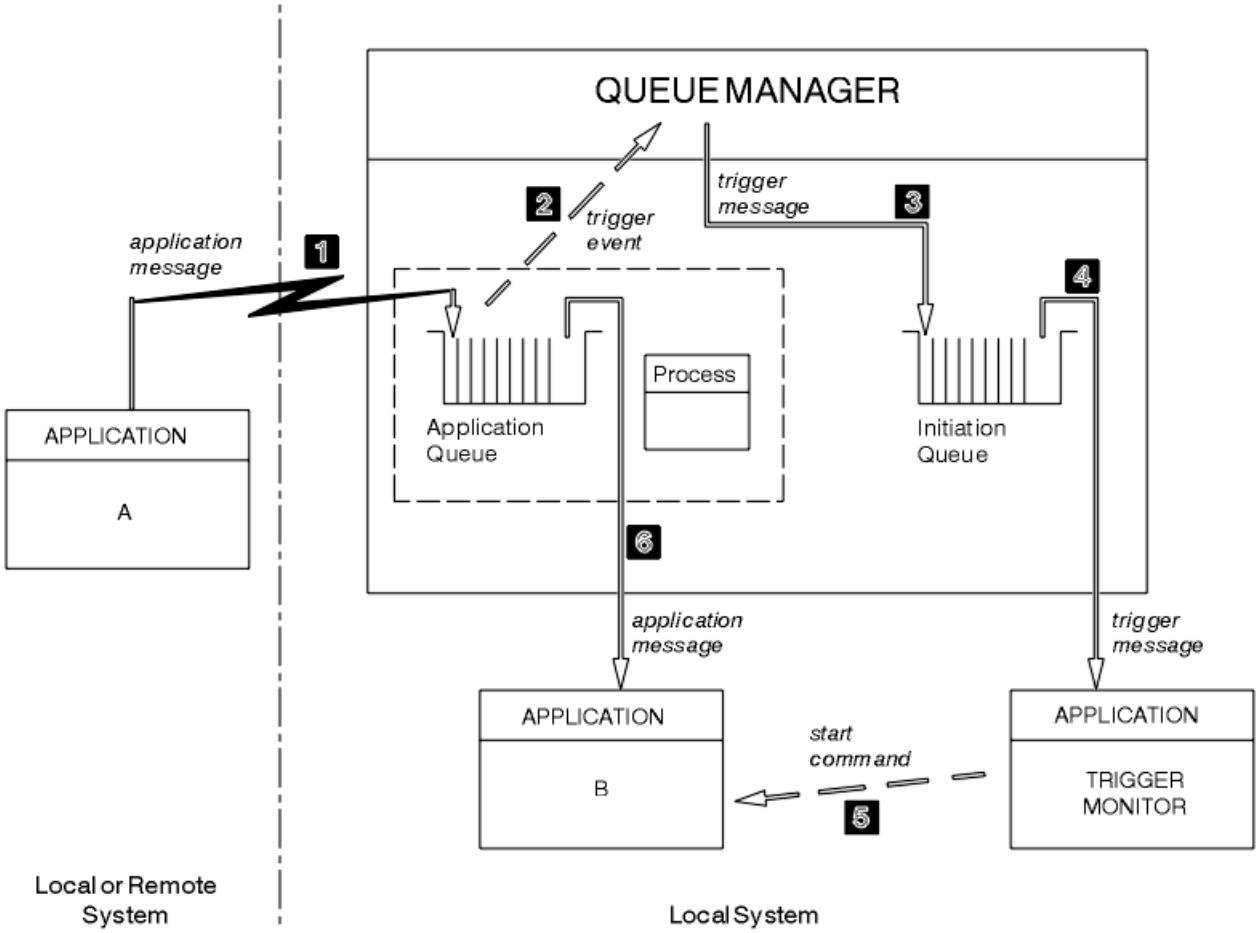
Na všech platformách je odpovědný za spouštění kanálů speciální monitor spouštěčů, který je znám jako iniciátor kanálu.

z/OS V systému z/OSje inicializátor kanálu obvykle spuštěn ručně nebo může být proveden automaticky při spuštění správce front změnou hodnoty CSQINP2 ve spouštěcím skriptu JCL správce front.

Multi V systému Multiplatformyje inicializátor kanálu spuštěn automaticky při spuštění správce front nebo jej lze spustit ručně pomocí příkazu **runmqchi** .

Další informace viz “Monitory spouštěčů” na stránce 842.

Chcete-li porozumět tomu, jak spouštění funguje, zvažte produkt Obrázek 105 na stránce 831, který je příkladem typu spouštěče FIRST (MQTT_FIRST).



Obrázek 105. Tok zpráv aplikace a spouštěče

V produktu Obrázek 105 na stránce 831je posloupnost událostí následující:

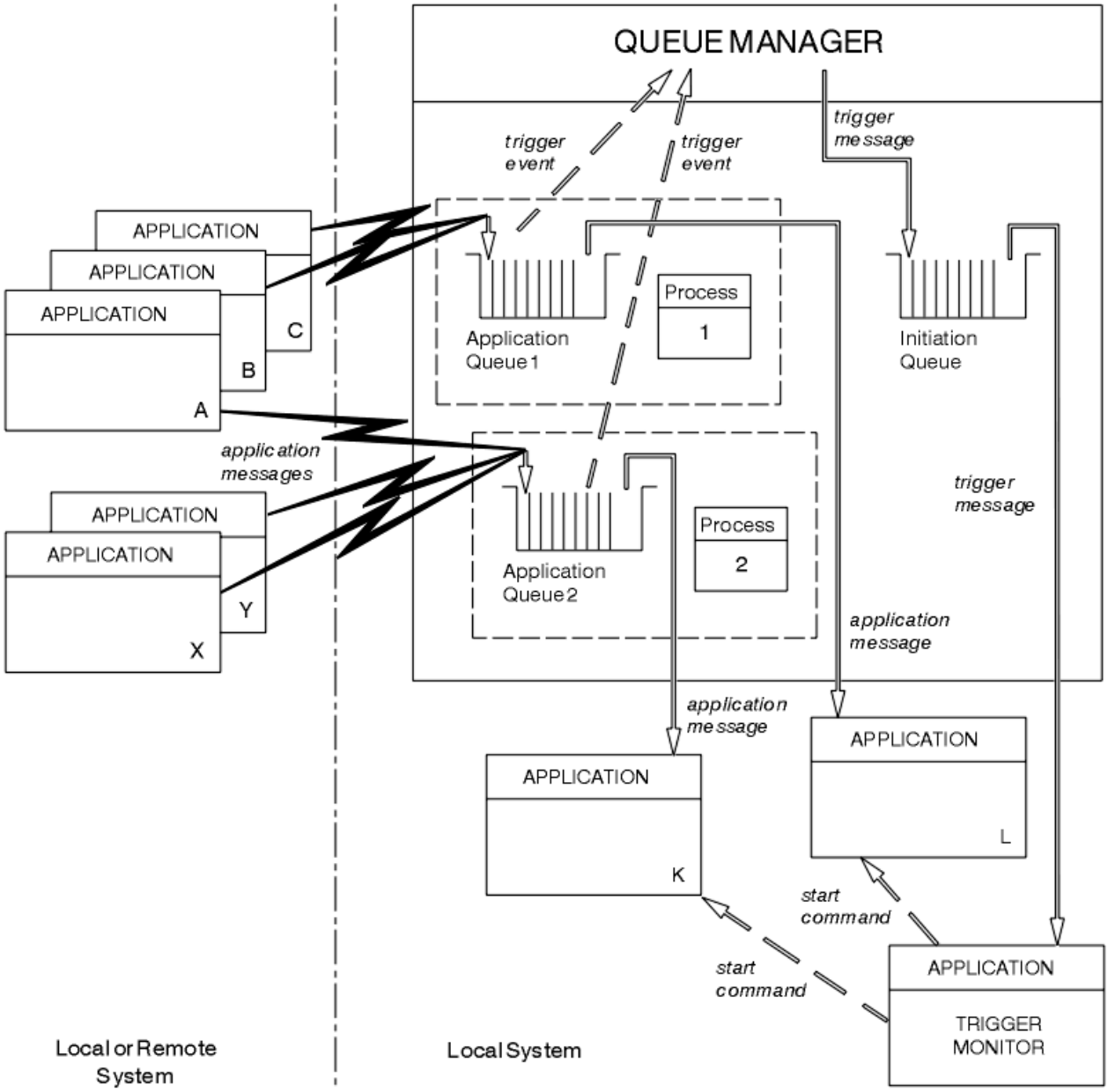
1. Aplikace A, která může být buď lokální, nebo vzdálená vzhledem ke správci front, vloží zprávu do fronty aplikací. Žádná aplikace nemá tuto frontu otevřenou pro vstup. Tato skutečnost je však relevantní pouze pro typ spouštěče FIRST a DEPTH.
2. Správce front zkontroluje, zda jsou splněny podmínky, za kterých musí generovat událost spouštěče. Jsou, a je generována událost spouštěče. Informace uchovávané v přidruženém objektu definice procesu se používají při vytváření zprávy spouštěče.
3. Správce front vytvoří zprávu spouštěče a vloží ji do inicializační fronty přidružené k této frontě aplikací, ale pouze v případě, že má aplikace (monitor spouštěčů) otevřenou vstupní frontu pro vstup.
4. Monitor spouštěčů načte zprávu spouštěče z inicializační fronty.
5. Monitor spouštěčů vydá příkaz pro spuštění aplikace B (serverová aplikace).

6. Aplikace B otevře aplikační frontu a načte zprávu.

Poznámka:

1. Je-li fronta aplikací otevřena pro vstup, libovolným programem a má spouštěcí sadu pro FIRST nebo DEPTH, nedojde k žádné události spouštěče, protože fronta je již obsluhována.
2. Není-li inicializační fronta otevřena pro vstup, správce front negeneruje žádné zprávy spouštěče. Bude čekat, dokud aplikace neotevře inicializační frontu pro vstup.
3. Při použití spouštěče pro kanály použijte typ spouštěče FIRST nebo DEPTH.
4. Spouštěné aplikace se spouštějí pod ID uživatele a skupinou uživatele, který spustil monitor spouštěčů, uživatele produktu CICS nebo uživatele, který spustil správce front.

Dosud byl vztah mezi frontami v rámci spuštění pouze na jednom z nich. Zvažte Obrázek 106 na stránce 832.



Obrázek 106. Vztah front v rámci spouštěče

Fronta aplikace má přidružený objekt definice procesu, který uchovává podrobnosti o aplikaci, která bude zpracovávat tuto zprávu. Správce front umístí informace do zprávy spouštěče tak, aby byla nutná pouze jedna inicializační fronta. Monitor spouštěčů extrahuje tyto informace ze zprávy spouštěče a spustí příslušnou aplikaci, aby se mohla vypořádat se zprávou na každé frontě aplikací.

Pamatujte na to, že pokud chcete spustit spuštění kanálu, nemusíte definovat objekt definice procesu. Definice přenosové fronty může určit, který kanál se má spustit.

Následující odkazy použijte k vyhledání dalších informací o spouštění aplikací produktu IBM MQ pomocí spouštěčů:

- "Nezbytné předpoklady pro spuštění" na stránce 833
- "Podmínky pro událost spouštěče" na stránce 835
- "Řízení událostí spouštěče" na stránce 839
- "Návrh aplikace, která používá spuštěné fronty" na stránce 841
- "Monitory spouštěčů" na stránce 842
- "Vlastnosti zpráv spouštěče" na stránce 845
- "Když spouštění nefunguje" na stránce 847

Související pojmy

"Přehled rozhraní fronty zpráv" na stránce 690
Zde jsou uvedeny informace o komponentách rozhraní MQI (Message Queue Interface).

"Připojování k správci front a odpojování od něj" na stránce 704
Chcete-li používat programovací služby IBM MQ , musí mít program připojení ke správci front. V této části se dozvíte, jak se připojit k správci front a odpojit je od správce front.

"Otevírání a zavírání objektů" na stránce 712
Tyto informace poskytují vhled do otevírání a zavírání objektů IBM MQ .

"Vložení zpráv do fronty" na stránce 723
V této části se dozvíte, jak vložit zprávy do fronty.

"Získávání zpráv z fronty" na stránce 737
Tyto informace použijte k získání informací o získávání zpráv z fronty.

"Inquaning about a nastavení atributů objektu" na stránce 815
Atributy jsou vlastnosti, které definují charakteristiky objektu IBM MQ .

"Potvrzení a zálohování jednotek práce" na stránce 818
Tyto informace popisují, jak potvrdit a zazálohovat všechny zotavitelné operace get a put, které se vyskytly v jednotce práce.

"Práce s MQI a klastry" na stránce 847
Existují speciální volby na voláních a návratových kódech, které se vztahují ke klastrování.

"Použití a zápis aplikací v systému IBM MQ for z/OS" na stránce 851
Aplikace produktu IBM MQ for z/OS mohou být vytvořeny z programů spuštěných v mnoha různých prostředích. To znamená, že mohou využít výhody zařízení dostupných ve více než jednom prostředí.

"Aplikace mostu IMS a IMS v systému IBM MQ for z/OS" na stránce 57
Tyto informace vám pomohou při psaní aplikací produktu IMS pomocí produktu IBM MQ.

Nezbytné předpoklady pro spuštění

Tyto informace použijte k seznámení se s kroky, které je třeba provést před použitím spouštěče.

Před tím, než bude vaše aplikace moci využívat spouštění, postupujte takto:

1. Proveďte jednu z následujících akcí:
   a. Vytvořte inicializační frontu pro frontu aplikací. Příklad:

```
DEFINE QLOCAL (initiation.queue) REPLACE      +
```

```
          LIKE (SYSTEM.DEFAULT.INITIATION.QUEUE)      +
          DESCR ('initiation queue description')
```

, nebo

b. Určete název lokální fronty, která existuje a může ji použít vaše aplikace (obvykle se jedná o název SYSTEM.DEFAULT.INITIATION.QUEUE nebo, pokud spouštíte kanály se spouštěči, SYSTEM.CHANNEL.INITQ) a zadejte její název do pole *InitiationQName* ve frontě aplikací.

2. Přidružte inicializační frontu k frontě aplikací. Správce front může vlastnit více než jednu inicializační frontu. Možná budete chtít, aby některé z vašich aplikačních front byly obsluhovány různými programy, v tom případě můžete použít jednu inicializační frontu pro každý obsluhující program, i když to nemusíte. Zde je příklad, jak vytvořit aplikační frontu:

```
DEFINE QLOCAL (application.queue) REPLACE   +
LIKE (SYSTEM.DEFAULT.LOCAL.QUEUE)           +
DESCR ('appl queue description')    +
INITQ ('initiation.queue')          +
PROCESS ('process.name')            +
TRIGGER                                     +
TRIGTYPE (FIRST)
```

IBM i Zde je výpis z CL programu pro IBM MQ for IBM i , který vytváří inicializační frontu:

```
/*   Queue used by AMQSINQA                                        */
            CRTMQMQ     QNAME('SYSTEM.SAMPLE.INQ')                  +
                        QTYPE(*LCL)  REPLACE(*YES)                  +
                        MQMNAME                                     +
                        TEXT('queue for AMQSINQA')                  +
                        SHARE(*YES)                /* Shareable   */+
                        DFTMSGPST(*YES)/* Persistent messages OK  */+
                                                                    +
                        TRGENBL(*YES)  /* Trigger control on      */+
                        TRGTYPE(*FIRST)/* Trigger on first message*/+
                        PRCNAME('SYSTEM.SAMPLE.INQPROCESS')         +
                        INITQNAME('SYSTEM.SAMPLE.TRIGGER')
```

3. Pokud spustíte aplikaci, vytvořte objekt definice procesu, který bude obsahovat informace vztahující se k aplikaci, která má sloužit ke zpracování vaší aplikační fronty. Chcete-li například spustit příkaz CICS payroll transakce s názvem PAYR, spusťte:

```
DEFINE PROCESS (process.name) +
       REPLACE +
       DESCR ('process description') +
       APPLICID ('PAYR') +
       APPLTYPE (CICS) +
       USERDATA ('Payroll data')
```

IBM i Zde je výpis z CL programu pro IBM MQ for IBM i , který vytváří objekt definice procesu:

```
/*   Process definition                                            */
            CRTMQMPRC   PRCNAME('SYSTEM.SAMPLE.INQPROCESS')         +
                        REPLACE(*YES)                               +
                        MQMNAME                                     +
                        TEXT('trigger process for AMQSINQA')        +
                        ENVDATA('JOBPTY(3)') /* Submit parameter  */+
                        APPID('AMQSINQA')    /* Program name      */
```

Když správce front vytvoří zprávu spouštěče, okopíruje informace z atributů objektu definice procesu do zprávy spouštěče.

Platforma	Vytvoření objektu definice procesu
Systémy UNIX, Linux, and Windows	Použijte atribut DEFINE PROCESS nebo použijte SYSTEM.DEFAULT.PROCESS a úprava pomocí příkazu ALTER PROCESS
z/OS z/OS z/OS	Použijte příkaz DEFINE PROCESS (viz ukázkový kód v kroku “3” na stránce 834), nebo použijte operace a ovládací panely.
IBM i IBM i IBM i	Použijte CL program obsahující kód jako v kroku “3” na stránce 834.

4. Volitelné: Vytvořte definici přenosové fronty a použijte prázdné místo pro atribut **ProcessName** .

   Atribut **TrigData** může obsahovat název kanálu, který má být spuštěn, nebo jej lze ponechat prázdný. S výjimkou IBM MQ for z/OS, pokud je ponechán prázdný, prohledává inicializátor kanálu soubory s definicemi kanálů, dokud nenajde kanál, který je přidružen k uvedené přenosové frontě. Když správce front vytvoří zprávu spouštěče, zkopíruje informace z atributu **TrigData** definice přenosové fronty do zprávy spouštěče.

5. Pokud jste vytvořili objekt definice procesu k určení vlastností aplikace, která slouží ke zpracování vaší aplikační fronty, přidružte objekt procesu k frontě aplikací tím, že ji pojmenujete v atributu **ProcessName** fronty.

Platforma	Použít příkazy
Systémy UNIX, Linux, and Windows	POZMĚNIT QLOCAL
z/OS z/OS z/OS	POZMĚNIT QLOCAL
IBM i IBM i IBM i	CHGMQMQ

6. Spouštět instance spouštěče IBM i (nebo spouštěcí servery v produktu IBM MQ for IBM i) , které slouží k obsluze inicializačních front, které jste definovali. Další informace viz “Monitory spouštěčů” na stránce 842.

Chcete-li si být vědomi jakýchkoli nedoručených zpráv spouštěče, ujistěte se, že má správce front definovanou frontu nedoručených zpráv (nedoručená zpráva). Do pole správce front produktu *DeadLetterQName* zadejte název fronty.

Poté můžete nastavit podmínky spouštěče, které vyžadujete, s použitím atributů objektu fronty, který definuje vaši frontu aplikací. Další informace viz “Řízení událostí spouštěče” na stránce 839.

Podmínky pro událost spouštěče

Správce front vytvoří zprávu spouštěče, jsou-li splněny podmínky uvedené v tomto tématu.

Odkazy na sdílené fronty v tomto tématu znamenají sdílené fronty v rámci skupiny sdílení front, které jsou k dispozici pouze v produktu IBM MQ for z/OS.

Následující podmínky způsobí, že správce front vytvoří zprávu spouštěče:

1. Zpráva je ve frontě *vložena* do fronty.
2. Zpráva má prioritu větší nebo rovnou prahové hodnotě triggeru prahové hodnoty fronty. Tato priorita je nastavena v atributu lokální fronty **TriggerMsgPriority** ; pokud je nastavena na nulu, libovolná zpráva se kvalifikuje.

3. Počet zpráv ve frontě s prioritou vyšší nebo rovnou *TriggerMsgPriority* byl dříve, v závislosti na *TriggerType*:
   - Nula (pro typ spouštěče MQTT_FIRST)
   - Libovolné číslo (pro typ spouštěče MQTT_EVERY)
   - *TriggerDepth* minus 1 (pro typ spouštěče MQTT_DEPTH)

   Poznámka:

   a. U nesdílených lokálních front počítá správce front jak potvrzené, tak nepotvrzené zprávy, když vyhodnocuje, zda existují podmínky pro událost spouštěče. Proto může být aplikace spuštěna, když nejsou k dispozici žádné zprávy k načtení, protože zprávy ve frontě nebyly potvrzeny. V této situaci zvažte použití volby čekání s vhodným prostorem *WaitInterval*, takže aplikace čeká na příchod svých zpráv.

   b. U lokálních sdílených front počítá správce front pouze potvrzené zprávy.

4. Pro spuštění typu FIRST nebo DEPTH žádný program nemá otevřenou frontu aplikace pro odstranění zpráv (tj. lokální atribut fronty **OpenInputCount** je nula).

   Poznámka:

   a. Pro sdílené fronty platí speciální podmínky, pokud více správců front má spuštěno monitorování spuštěných proti frontě. V této situaci, pokud má jeden nebo více správců front otevřenou frontu pro sdílený vstup, jsou kritéria spouštěcího impulsu na ostatních správcích front považována za *TriggerType* MQTT_FIRST a *TriggerMsgPriority* nula. Když všechny správce front zavřou frontu pro vstup, obnoví se podmínky spouštěče na podmínky uvedené v definici fronty.

      Ukázkový scénář, který je ovlivněn touto podmínkou, je více správců front QM1, QM2a QM3 s monitorem spouštěčů spuštěným pro aplikační frontu A. Zpráva dorazí na A splňující podmínky pro spuštění a v inicializační frontě se vygeneruje zpráva spouštěče. Monitor spouštěčů na systému QM1 získá zprávu spouštěče a spustí aplikaci. Spuštěná aplikace otevře aplikační frontu pro sdílený vstup. Od tohoto bodu za spouštěcí podmínky pro frontu aplikací A se vyhodnotí jako *TriggerType* MQTT_FIRST a *TriggerMsgPriority* na správci front QM2 a QM3, dokud QM1 uzavře frontu aplikací.

   b. Pro sdílené fronty je tato podmínka použita pro každého správce front. To znamená, že *OpenInputCount* správce front pro frontu musí být nula pro zprávu spouštěče, která má být generována pro danou frontu daným správcem front. Pokud však kterýkoli správce front ve skupině sdílení front má otevřenou frontu pomocí volby MQOO_INPUT_EXCLUSIVE, nebude pro danou frontu vygenerována žádná zpráva spouštěče žádné ze správců front v rámci skupiny sdílení front.

      Změna způsobu vyhodnocení podmínek spouštěče se vyskytne, když spuštěná aplikace otevře frontu pro vstup. Ve scénářích, kde je spuštěný pouze jeden monitor spouštěčů, mohou mít jiné aplikace stejný účinek, protože podobně otevírají vstupní frontu pro vstup. Nezáleží na tom, zda byla aplikační fronta otevřena aplikací spuštěnou monitorem spouštěčů, nebo jinou aplikací; jedná se o skutečnost, že je fronta otevřena pro vstup na jiném správci front, který způsobuje změnu kritérií spouštěče.

5. On IBM MQ for z/OS, if the application queue is one with a **Usage** attribute of MQUS_NORMAL, get requests for it are not inhibited (that is, the **InhibitGet** queue attribute is MQQA_GET_ALLOWED). Je-li fronta aplikací spuštěna také s atributem **Usage** MQUS_XMITQ, nebudou mít k dispozici žádné požadavky, pro které nejsou blokovány žádné požadavky.
6. Proveďte jednu z následujících akcí:
   - Atribut lokální fronty **ProcessName** pro danou frontu není prázdný a byl vytvořen objekt definice procesu identifikovaný tímto atributem, nebo
   - Atribut lokální fronty **ProcessName** pro danou frontu je prázdný, ale fronta je přenosová fronta. Protože je definice procesu volitelná, může atribut **TriggerData** obsahovat také název kanálu, který má být spuštěn. V takovém případě bude zpráva spouštěče obsahovat atributy s následujícími hodnotami:

- **QName**: název fronty
- **ProcessName**: mezery
- **TriggerData**: data spouštěče
- **ApplType**: MQAT_UNKNOWN
- **ApplId**: mezery
- **EnvData**: mezery
- **UserData**: mezery

7. Inicializační fronta byla vytvořena a byla zadána v atributu lokální fronty **InitiationQName** . Dále:
   - Požadavky na získání nejsou blokovány pro inicializační frontu (to znamená, že hodnota atributu fronty **InhibitGet** je MQQA_GETALLOWED).
   - Požadavky PUT nesmí být blokovány pro inicializační frontu (to znamená, že hodnota atributu fronty **InhibitPut** musí být MQQA_PUT_ALLOWED).
   - Hodnota atributu **Usage** inicializační fronty musí být MQUS_NORMAL.
   - V prostředích, ve kterých jsou podporovány dynamické fronty, nesmí být inicializační fronta dynamická fronta, která byla označena jako logicky odstraněná.
8. Monitor spouštěčů má v současné době inicializační frontu otevřenou pro odstranění zpráv (to znamená, že lokální atribut fronty **OpenInputCount** je větší než nula).
9. Ovládací prvek spouštěče (atribut lokální fronty**TriggerControl**) pro frontu aplikací je nastaven na hodnotu MQTC_ON. Chcete-li to provést, nastavte atribut **trigger** při definování fronty nebo použijte příkaz ALTER QLOCAL.
10. Typ spouštěče (lokální atribut fronty**TriggerType**) není MQTT_NONE.

    Pokud jsou splněné všechny požadované podmínky a zpráva, která způsobila podmínku spouštěče, je vložena jako část pracovní jednotky, zpráva spouštěcího impulsu se nestane dostupnou pro načtení aplikací monitoru spouštěčů, dokud nebude dokončena jednotka práce, zda je jednotka práce potvrzena, nebo, pro typ triggeru MQTT_FIRST nebo MQTT_DEPTH, zálohováno.
11. Vhodná zpráva se umístí do fronty pro **TriggerType** MQTT_FIRST nebo MQTT_DEPTH a do fronty:
    - Nebyl dříve prázdný (MQTT_FIRST) nebo
    - Měl **TriggerDepth** nebo více zpráv (MQTT_DEPTH)

    a podmínky “2” na stránce 835 až “10” na stránce 837 (kromě “3” na stránce 836) jsou splněny, pokud je v případě MQTT_FIRST dostatečný interval (atribut správce front**TriggerInterval**), uplynulo od doby, kdy byla pro tuto frontu zapsána poslední zpráva spouštěče.

    Toto je povolení pro server fronty, který se ukončí před zpracováním všech zpráv ve frontě. Účelem intervalu spouštěče je snížit počet vygenerovaných duplicitních zpráv spouštěče.

    Poznámka: Pokud zastavíte a znovu spustíte správce front, časovač **TriggerInterval** se znovu nastaví. Je zde malé okno, v jehož průběhu je možné vytvořit dvě zprávy spouštěče. Okno existuje, když je atribut spouštěče fronty nastaven tak, aby byl povolen ve stejnou dobu jako zpráva přijde a fronta nebyla dříve prázdná (MQTT_FIRST) nebo měla **TriggerDepth** nebo více zpráv (MQTT_DEPTH).
12. Jediná aplikace obsluhující frontu nabízí volání MQCLOSE, pro **TriggerType** MQTT_FIRST nebo MQTT_DEPTH, a je zde alespoň:
    - jeden (MQTT_FIRST) nebo
    - **TriggerDepth** (MQTT_DEPTH)

    jsou také zprávy ve frontě s dostatečnou prioritou (podmínka “2” na stránce 835) a podmínky “6” na stránce 836 až “10” na stránce 837 jsou také splněny.

    Toto je povolení pro server fronty, který vydává volání MQGET, vyhledá frontu prázdnou, a tak skončí; avšak v intervalu mezi voláními MQGET a MQCLOSE je doručena jedna nebo více zpráv.

    Poznámka:

a. Pokud program obsluhující aplikační frontu nenačte všechny zprávy, může to způsobit zavřenou smyčku. Pokaždé, když program zavře frontu, vytvoří správce front další spouštěcí zprávu, která způsobí, že monitor spouštěčů znovu spustí program serveru.

b. Pokud program obsluhující frontu aplikací zálohuje svůj požadavek na získání (nebo pokud se program ukončí) před zavřením fronty, stane se totéž. Pokud však program frontu uzavře předtím, než dojde k získání požadavku na získání, a fronta je jinak prázdná, nebude vytvořena žádná zpráva spouštěče.

c. Chcete-li zabránit výskytu takové smyčky, použijte pole *BackoutCount* MQMD k detekci zpráv, které jsou opakovaně vráceny. Další informace viz “Zprávy, které jsou zálohovány” na stránce 40.

13. Pomocí MQSET nebo příkazu jsou splněny následující podmínky:

a. • **TriggerControl** se změní na MQTC_ON, nebo

• **TriggerControl** je již MQTC_ON a hodnota **TriggerType**, **TriggerMsgPriority**nebo **TriggerDepth** (je-li relevantní) se změní,

a je zde alespoň:

• jeden (MQTT_FIRST nebo MQTT_EVERY), nebo

• **TriggerDepth** (MQTT_DEPTH)

zprávy ve frontě s dostatečnou prioritou (podmínka “2” na stránce 835), a podmínky “4” na stránce 836 až “10” na stránce 837 (kromě “8” na stránce 837) jsou také splněny.

To umožňuje aplikaci nebo operátorovi, který mění spouštěcí kritéria, když již jsou splněny podmínky pro výskyt spouštěče.

b. Hodnota atributu fronty **InhibitPut** inicializační fronty se změní z hodnoty MQQA_PUT_INHIBITED na MQQA_PUT_ALLOWED a je alespoň:

• jeden (MQTT_FIRST nebo MQTT_EVERY), nebo

• **TriggerDepth** (MQTT_DEPTH)

zprávy s dostatečnou prioritou (podmínka “2” na stránce 835) ve všech frontách, pro které se jedná o inicializační frontu, jsou také splněny podmínky “4” na stránce 836 až “10” na stránce 837 . (Jedna zpráva spouštěče je generována pro každou takovou frontu splňující podmínky.)

To umožňuje, aby se zprávy spouštěče negenerovaly kvůli podmínce MQQA_PUT_INHIBITED na inicializační frontě, ale tento stav se nyní změnil.

c. Hodnota atributu fronty **InhibitGet** fronty aplikací se změní z hodnoty MQQA_GET_INHIBITED na MQQA_GET_ALLOWED a je alespoň:

• jeden (MQTT_FIRST nebo MQTT_EVERY), nebo

• **TriggerDepth** (MQTT_DEPTH)

zprávy s dostatečnou prioritou (podmínka “2” na stránce 835) na frontě a podmínky “4” na stránce 836 až “10” na stránce 837, kromě “5” na stránce 836, jsou také splněny.

To umožňuje, aby aplikace byly spuštěny pouze tehdy, když mohou načítat zprávy z aplikační fronty.

d. Aplikace monitoru spouštěčů vydá volání MQOPEN pro vstup z inicializační fronty a je zde alespoň:

• jeden (MQTT_FIRST nebo MQTT_EVERY), nebo

• **TriggerDepth** (MQTT_DEPTH)

zprávy s dostatečnou prioritou (podmínka “2” na stránce 835) ve všech frontách aplikací, pro které se jedná o inicializační frontu, a podmínky “4” na stránce 836 až “10” na stránce 837 (kromě “8” na stránce 837) jsou také spokojeni a žádná jiná aplikace nemá vstupní frontu otevřenou pro vstup (jedna zpráva spouštěče je generována pro každou takovou frontu splňující podmínky).

To znamená povolit zprávy přicházející do front, zatímco monitor spouštěčů není spuštěný, a pro restartování správce front a zprávy triggeru (které jsou přechodné), které se ztratí.

14. MSGDLVSQ je nastavena správně. Nastavíte-li parametr MSGDLVSQ=FIFO, budou zprávy doručovány do fronty nejprve v První odchozí zprávě. Priorita zprávy je ignorována a výchozí priorita fronty je přiřazena ke zprávě. Pokud je parametr **TriggerMsgPriority** nastaven na vyšší hodnotu, než je výchozí priorita fronty, nespustí se žádné zprávy. Je-li parametr **TriggerMsgPriority** nastaven na hodnotu rovnou nebo nižší než výchozí priorita fronty, spustí se spuštění typu FIRST, EVERY a DEPTH. Informace o těchto typech naleznete v popisu pole **TriggerType** v části "Řízení událostí spouštěče" na stránce 839.

    Pokud nastavíte MSGDLVSQ=PRIORITY a priorita zprávy je rovna nebo větší než pole *TriggerMsgPriority* , zprávy se počítají pouze ke spouštěcí události. V takovém případě se spustí spuštění typu FIRST, EVERY a DEPTH. Zadáte-li například 100 zpráv nižší priority než **TriggerMsgPriority**, efektivní hloubka fronty pro účely spuštění je stále nula. Pokud poté vložíte další zprávu do fronty, ale tentokrát je priorita větší nebo rovna hodnotě **TriggerMsgPriority**, bude se efektivní hloubka fronty zvětšovat od nuly do jedné a podmínka pro **TriggerType** FIRST je splněna.

Poznámka:

1. Z kroku "12" na stránce 837 (kde jsou zprávy spouštěče generovány jako výsledek některé události, která není doručena do fronty aplikací), se zpráva spouštěče nevloží jako součást pracovní jednotky. Pokud je **TriggerType** MQTT_EVERY a je-li ve frontě aplikace jedna nebo více zpráv, vygeneruje se pouze jedna zpráva spouštěče.
2. Pokud objekt IBM MQ segmentuje zprávu během operace MQPUT, událost triggeru nebude zpracována, dokud nebudou všechny segmenty úspěšně umístěny do fronty. Jakmile se však segmenty zprávy nacházejí ve frontě, produkt IBM MQ je považuje za jednotlivé zprávy pro účely spouštěče. Například jedna logická zpráva rozdělená na tři kusy způsobí zpracování pouze jedné události triggeru, když je první MQPUT a segmentovaná. Avšak každý ze tří segmentů způsobuje, že jejich vlastní události triggeru budou zpracovány, protože jsou přesunuty přes síť IBM MQ .

Řízení událostí spouštěče

Události spouštěče můžete řídit pomocí některých atributů, které definují frontu aplikací. Tyto informace také uvádí příklady použití typů spouštěčů: EVERY, FIRST a DEPTH.

Můžete povolit nebo zakázat spouštění a můžete vybrat počet nebo prioritu zpráv, které se počítají do události spouštěče. V části Atributy objektůje uveden úplný popis těchto atributů.

Příslušné atributy jsou:

TriggerControl
: Tento atribut použijte k povolení a zakázání spouštění pro aplikační frontu.

TriggerMsgPriority
: Minimální priorita, kterou musí zpráva mít, aby se napočítala ke spouštěcí události. Pokud je zpráva s prioritou nižší než *TriggerMsgPriority* doručena do fronty aplikací, správce front zprávu ignoruje, pokud určí, zda má být vytvořena zpráva spouštěče. Je-li parametr *TriggerMsgPriority* nastaven na hodnotu nula, budou všechny zprávy započítávají do události spouštěče.

TriggerType
: Kromě spouštěče typu NONE (který zakazuje spouštění stejně jako nastavení *TriggerControl* na OFF), můžete použít následující typy spouštěčů k nastavení citlivosti fronty na události triggeru:

  Každý
  : Událost spouštěče se vyskytne pokaždé, když přijde zpráva do aplikační fronty. Tento typ spouštěče použijte v případě, že chcete spustit více instancí aplikace.

  PRVNÍ
  : Událost spouštěče se vyskytne pouze, když se počet zpráv v aplikační frontě změní z nuly na jeden. Tento typ spouštěče použijte, pokud chcete, aby se obsluhující program spustil při příchodu první zprávy do fronty, pokračovat až do doby, kdy nebudou další zprávy ke zpracování, pak ukončete. Frontu musíte vždy zpracovat, dokud nebude prázdná. Další informace najdete v tématu "Speciální případ typu spouštěče FIRST" na stránce 840.

DEPTH
Událost spouštěče se vyskytuje pouze tehdy, když se počet zpráv ve frontě aplikace dosáhne hodnoty atributu **TriggerDepth** . Typickým použitím tohoto typu spouštění je spuštění programu, když jsou přijaty všechny odpovědi na sadu požadavků.

Spouštění podle hloubky: Se spuštěním hloubkou zakáže správce front spouštění (pomocí atributu *TriggerControl*) poté, co vytvoří zprávu spouštěče. Vaše aplikace musí po provedení této události znovu povolit spuštění spouštěče (pomocí volání MQSET).

Akce zakázání spouštěče není pod řízením synchronizačního bodu, takže spuštění nelze znovu povolit pomocí zálohování jednotky práce. Pokud program zálohuje požadavek na vložení, který způsobil událost spouštěče, nebo pokud se program ukončí, musíte znovu povolit spouštění pomocí volání MQSET nebo příkazu ALTER QLOCAL.

TriggerDepth
Počet zpráv ve frontě, které způsobí událost spouštěče při použití spouštěče podle hloubky.

Podmínky, které musí být splněny pro vytvoření zprávy spouštěče pro správce front, jsou popsány v tématu "Podmínky pro událost spouštěče" na stránce 835.

Příklad použití typu spouštěče EVERY

Představte si aplikaci, která generuje požadavky na pojištění motorových vozidel. Aplikace může odeslat zprávy požadavku na celou řadu pojišťoven a pokaždé uvést stejnou odpověď do fronty. Může nastavit spouštěč typu KAŽDÉ v této odpovědi-na frontu tak, aby pokaždé, když přijde odpověď, mohla odpověď spustit instanci serveru pro zpracování odpovědi.

Příklad použití typu spouštěče FIRST

Zamysleme se nad organizací s mnoha pobočkami, které každý předává podrobnosti o pracovních dnech do hlavní kanceláře. Všichni to dělají ve stejnou dobu, na konci pracovního dne, a v kanceláři ústředí je aplikace, která zpracovává podrobnosti ze všech poboček. První zpráva, která má přijet do hlavní kanceláře, může způsobit událost spouštěče, která spustí tuto aplikaci. Tato aplikace by pokračovala ve zpracování, dokud ve frontě nejsou žádné další zprávy.

Příklad použití typu spouštěče DEPTH

Vezměme si cestovní kancelář aplikace, která vytváří jeden požadavek na potvrzení letenky rezervaci, potvrdit rezervaci pro hotelový pokoj, pronájem auta, a objednání některých cestujících kontroly. Aplikace může tyto položky rozdělit do čtyř zpráv vzniklých při zpracování požadavků, přičemž každá z nich bude posílat do samostatného místa určení. Může nastavit spouštěč typu DEPTH ve své odpovědi na frontu (s hloubkou nastaveným na hodnotu 4), takže bude restartována pouze v případě, že dorazily všechny čtyři odpovědi.

Pokud další zpráva (pravděpodobně z jiného požadavku) dorazí do fronty pro odpovědi před posledním ze čtyř odpovědí, je žádající aplikace spuštěna předčasně. Chcete-li se vyhnout tomu, že při použití funkce DEPTH ke shromažďování více odpovědí na požadavek, vždy pro každý požadavek použijte novou frontu pro odpověď.

Speciální případ typu spouštěče FIRST

S typem spouštěče FIRST, pokud je zpráva ve frontě zpráv již doručena, když přijde jiná zpráva, správce front obvykle nevytvoří další zprávu spouštěče.

Avšak aplikace obsluhující frontu nemusí ve skutečnosti otevřít frontu (například, aplikace by mohla skončit, možná kvůli problému se systémem). Pokud bylo do objektu definice procesu zadáno nesprávné jméno aplikace, aplikace obsluhující frontu nevyzvedne žádnou ze zpráv. Pokud v těchto situacích přijde do fronty aplikace jiná zpráva, není spuštěn žádný server ke zpracování této zprávy (a žádné další zprávy ve frontě).

K vyřešení tohoto stavu vytvoří správce front další zprávy spouštěče za následujících okolností:

- Pokud do fronty aplikací dorazí další zpráva, ale pouze v případě, že od správce front je vytvořena poslední spouštěcí zpráva pro danou frontu, pouze v případě, že uplynul předdefinovaný časový interval. Tento časový interval je definován v atributu správce front *TriggerInterval*. Jeho výchozí hodnota je 999 999 999 milisekund.
- V systému IBM MQ for z/OSjsou fronty aplikací, které pojmenují otevřenou inicializační frontu, pravidelně skenovány. Pokud *TRIGINT* milisekund uplynuly od odeslání poslední zprávy spouštěče a fronta splňuje podmínky pro událost triggeru a CURDEPTH je větší než nula, vygeneruje se zpráva spouštěče. Tento proces se nazývá backstop spouštějící.

Při rozhodování o hodnotě intervalu spouštěče, který má být použit ve vaší aplikaci, zvažte následující body:

- Pokud jste nastavili *TriggerInterval* na nízkou hodnotu a neexistuje žádná aplikace obsluhující aplikační frontu, typ spouštěče FIRST se může chovat jako typ spouštěče KAŽDÉ. Závisí na rychlosti, jakou jsou zprávy vloženy do aplikační fronty, která naopak může záviset na jiné aktivitě systému. Je tomu tak proto, že je-li interval spouštěče velmi malý, generuje se při každém vložení zprávy do aplikační fronty jiná zpráva spouštěče i přesto, že typ spouštěče je FIRST, nikoli KAŽDÉ. (Typ spouštěče FIRST s intervalem spouštěče nula je ekvivalentní typu spouštěče ECVERY.)
- Pokud na IBM MQ for z/OS nastavíte *TRIGINT* na nízkou hodnotu a neexistuje žádná aplikace obsluhující typ spouštěče PRVNÍ, spustí se spouštěcí program backstop zprávu spouštěče pokaždé, když dojde k periodickému skenování front aplikací, které pojmenují otevřené fronty inicializace.
- Je-li zálohována jednotka práce (viz Spouštěcí zprávy a jednotky práce) a interval triggeru byl nastaven na vysokou hodnotu (nebo výchozí hodnotu), jedna zpráva spouštěče se vygeneruje, když je jednotka práce zálohována. Avšak, pokud jste nastavili interval spouštěče na nízkou hodnotu nebo na nulu (způsobuje typ spouštěče FIRST, aby se choval jako typ spouštěče EVERY), lze generovat mnoho zpráv spouštěče. Je-li jednotka práce zálohována, všechny zprávy spouštěče jsou stále k dispozici. Počet vygenerovaných zpráv spouštěče závisí na intervalu spouštěče. Je-li interval triggeru nastaven na nulu, vygeneruje se maximální počet zpráv.

Návrh aplikace, která používá spuštěné fronty

Nyní jste viděli, jak nastavit a řídit spouštění vašich aplikací. Zde je několik tipů, které byste měli zvážit při návrhu vaší aplikace.

Zprávy spouštěče a jednotky práce

Zprávy triggeru vytvořené z důvodu událostí triggeru, které nejsou součástí jednotky práce, se umístí do inicializační fronty, mimo jakoukoli jednotku práce, bez závislosti na jakýchkoli jiných zprávách a jsou k dispozici pro načtení okamžitě monitorem spouštěče.

Zprávy spouštěče vytvořené z důvodu událostí triggeru, které jsou součástí jednotky práce, jsou zpřístupněny na inicializační frontě při vyřešení transakce UOW, ať už je jednotka práce potvrzená nebo zálohovaná

Pokud správce front nevloží do inicializační fronty zprávu spouštěče, bude umístěna do fronty nedoručených zpráv (undelivered-message).

Poznámka:

1. Správce front počítá potvrzené i nepotvrzené zprávy, když vyhodnocuje, zda existují podmínky pro událost spouštěče.

   S spuštěním typu FIRST nebo DEPTH jsou zprávy spouštěče zpřístupněny, i když je jednotka práce zálohována tak, že je zpráva spouštěče vždy dostupná, když jsou splněny požadované podmínky. Předpokládejme například, že požadavek na vložení do jednotky práce pro frontu, která je spuštěna se spouštěčem typu FIRST. To způsobí, že správce front vytvoří zprávu spouštěče. Pokud se vyskytne jiný požadavek na vložení z jiné pracovní jednotky, nezpůsobí to jinou událost spouštěče, protože se počet zpráv ve frontě aplikace nyní změnil z jednoho na dva, což nesplňuje podmínky pro událost triggeru. Nyní, když je zálohována první jednotka práce, ale druhá je potvrzena, je stále vytvořena zpráva spouštěče.

To znamená, že zprávy spouštěče se někdy vytvoří, když nejsou splněny podmínky pro událost triggeru. Aplikace, které používají spouštěče, musí být vždy připraveny ke zpracování této situace. Doporučuje se použít volbu čekání s voláním MQGET a nastavit hodnotu *WaitInterval* na vhodnou hodnotu.

Vytvořené zprávy spouštěče jsou vždy k dispozici, ať už je jednotka práce zálohována nebo potvrzena.

2. Pro lokální sdílené fronty (tj. sdílené fronty ve skupině sdílení front) správce front počítá pouze potvrzené zprávy.

Získávání zpráv ze spuštěné fronty

Když navrhujete aplikace, které používají spouštěcí impuls, uvědomte si, že může existovat prodleva mezi monitorem spouštěčů spouštějícím program a dalšími zprávami, které budou k dispozici ve frontě aplikací. To se může stát, když se zpráva, která způsobí událost triggeru, potvrdí před ostatními.

Chcete-li povolit příchod zpráv, vždy použijte volbu wait, když používáte volání MQGET k odebrání zpráv z fronty, pro které jsou nastaveny podmínky spouštěče. Hodnota *WaitInterval* musí být dostatečná k tomu, aby mohla být nejdelší vhodná doba mezi vkládanou zprávou a potvrzenými potvrzenými volání. Pokud se zpráva dostaví ze vzdáleného správce front, bude tato doba ovlivněna:

- Počet zpráv, které byly vloženy před potvrzením
- Rychlost a dostupnost komunikačního spojení
- Velikosti zpráv

Příklad situace, kdy byste měli použít volání MQGET s volbou wait, považujte za stejný příklad, jaký jsme použili při popisování jednotek práce. Toto byl požadavek na vložení do jednotky práce pro frontu spuštěnou s typem spouštěče FIRST. Tato událost způsobí, že správce front vytvoří zprávu spouštěče. Pokud se vyskytne jiný požadavek na vložení z jiné pracovní jednotky, nezpůsobí to jinou událost triggeru, protože se počet zpráv ve frontě aplikací nezměnil od nuly na jeden. Nyní, když je zálohována první jednotka práce, ale druhá je potvrzena, je stále vytvořena zpráva spouštěče. Zpráva spouštěče se tedy vytvoří v době, kdy je zálohována první jednotka práce. Pokud dojde k závažnému zpoždění před potvrzením druhé zprávy, může pro ni být spuštěna spuštěná aplikace.

Při spuštění typu DEPTH může dojít k prodlevě i v případě, kdy jsou všechny relevantní zprávy nakonec potvrzeny. Předpokládejme, že atribut fronty **TriggerDepth** má hodnotu 2. Když dvě zprávy dorazí do fronty, druhá způsobí, že se vytvoří zpráva spouštěče. Je-li však druhá zpráva první zpráva, která má být potvrzena, je v té době k dispozici zpráva spouštěče. Monitor spouštěčů spustí program serveru, ale program může načíst pouze druhou zprávu, dokud nebude potvrzena první zpráva. Je tedy možné, že program bude muset čekat na zpřístupnění první zprávy.

Navrhněte aplikaci tak, aby byla ukončena, pokud nejsou k dispozici žádné zprávy pro načtení v době, kdy vyprší interval čekání. Pokud se jedna nebo více zpráv dostaví později, spoléhejte se na to, že se vaše aplikace bude zpracovávat. Tato metoda zabraňuje tomu, aby aplikace byly nečinné a zbytečně využívám prostředky.

Monitory spouštěčů

Pro správce front je monitor spouštěčů stejně jako jakákoli jiná aplikace, která obsluhuje frontu. Avšak, monitor spouštěčů obsluhuje inicializační fronty.

Monitor spouštěčů je obvykle spouštěný program. Když zpráva spouštěče dorazí do inicializační fronty, monitor spouštěčů tuto zprávu načte. Používá informace ve zprávě k vydání příkazu ke spuštění aplikace, která má zpracovat zprávy ve frontě aplikací.

Monitor spouštěčů musí předat dostatečné informace programu, že se spouští, aby mohl program provádět správné akce ve správné aplikační frontě.

Inicializátor kanálu je příkladem speciálního typu monitoru spouštěčů pro agenty kanálu zpráv. V této situaci však musíte použít buď typ spouštěče FIRST nebo DEPTH.

Windows UNIX *Spouštět monitory na systémech UNIX a Windows*

Toto téma obsahuje informace o monitorech spouštěčů, které jsou k dispozici v systémech UNIX a Windows .

Pro prostředí serveru jsou k dispozici následující monitory spouštěčů:

amqstrg0
Toto je ukázkový monitor spouštěčů, který poskytuje část funkce poskytované příkazem **runmqtrm**. Další informace o parametru amqstrg0viz “Použití ukázkových programů na více platformách” na stránce 1035 .

runmqtrm
Syntaxe tohoto příkazu je **runmqtrm** [*-m QMgrName*] [*-q InitQ*], kde QMgrName je správce front a InitQ je inicializační fronta. Výchozí fronta je SYSTEM.DEFAULT.INITIATION.QUEUE ve výchozím správci front. Vyvolá programy pro odpovídající zprávy spouštěče. Tento monitor spouštěčů podporuje výchozí typ aplikace.

Příkazový řetězec předaný monitorem spouštěčů do operačního systému je sestaven takto:

1. *ApplId* z příslušné definice PROCESS (je-li vytvořena)
2. Struktura MQTMC2 ohraničená dvojitými uvozovkami
3. *EnvData* z příslušné definice PROCESS (je-li vytvořena)

kde *ApplId* je jméno programu, který má být spuštěn, jak by byl zadán na příkazovém řádku.

Předaný parametr je struktura znaků MQTMC2 . Je vyvolán příkazový řetězec, který má tento řetězec přesně tak, jak je uveden ve dvojitých uvozovkách, aby příkaz systému akceptoval tento řetězec jako jeden parametr.

Monitor spouštěčů se nepodívá, zda se v inicializační frontě nachází jiná zpráva až do dokončení právě spuštěného aplikačního serveru. Pokud má aplikace mnoho práce, monitor spouštěčů nemusí být schopen držet krok s počtem příchozích zpráv, které přicházejí do systému. Máte dvě možnosti:

- Mají spuštěné více monitorů spouštěčů
- Spustit spuštěné aplikace na pozadí

Pokud máte spuštěnu více monitorů spouštěčů, můžete řídit maximální počet aplikací, které mohou být spuštěny v libovolném okamžiku. Spustíte-li aplikace na pozadí, IBM MQ na počet aplikací, které lze spustit, není vynucen žádný omezení.

Chcete-li spustit spuštěnou aplikaci na pozadí v systémech Windows , zadejte do pole *ApplId* předponu názvu vaší aplikace pomocí příkazu START. Příklad:

```
START ?B AMQSECHA
```

UNIX Chcete-li spustit spuštěnou aplikaci na pozadí v systému UNIX, vložte & na konec definice *EnvData* definice PROCESS.

Poznámka: Windows Pokud cesta Windows obsahuje mezery jako část názvu cesty, měla by být uzavřena v uvozovkách (") aby bylo zajištěno, že se s ním zachází jako s jedním argumentem. Příklad: "C:\Program Files\Application Directory\Application.exe".

Níže je uveden příklad řetězce APPLICID, kde název souboru obsahuje mezery jako část cesty:

```
START "" /B "C:\Program Files\Application Directory\Application.exe"
```

Syntaxe příkazu Windows START v příkladu obsahuje prázdný řetězec uzavřený ve dvojitých uvozovkách. Příkaz START určuje, že se první argument v uvozovkách bude považovat za titulek nového příkazu. Chcete-li se ujistit, že produkt Windows neudělá chybu v cestě aplikace pro argument 'title', přidejte před název aplikace řetězec názvu uzavřený do uvozovek do dvojitých uvozovek.

Pro klienta IBM MQ jsou k dispozici následující monitory spouštěčů:

runmqtmc
To je stejné jako runmqtrm s tím rozdílem, že obsahuje propojení s knihovnami produktu IBM MQ MQI client .

Monitor spouštěčů pro CICS
Monitor spouštěčů amqltmc0 je poskytován pro produkt CICS. Funguje stejným způsobem jako standardní monitor spouštěčů, runmqtrm, ale spustíte jej jiným způsobem a spouští CICS transakce.

Toto téma se vztahuje pouze na systémy Windows, UNIXa Linux x86-64 .

Je dodáván jako program CICS ; definujte jej se 4místným názvem transakce. Zadejte 4znakový název, abyste spustili monitor spouštěčů. Používá výchozího správce front (jak je pojmenováno v souboru qm.ini nebo, v systému IBM MQ for Windows, v registru) a v SYSTEM.CICS.INITIATION.QUEUE.

Chcete-li použít jiného správce front nebo frontu, sestavte strukturu monitoru spouštěčů MQTMC2 : to vyžaduje, abyste zapsali program pomocí volání EXEC CICS START, protože struktura je příliš dlouhá a nelze ji přidat jako parametr. Poté předejte strukturu MQTMC2 jako data do požadavku START pro monitor spouštěčů.

Použijete-li strukturu MQTMC2 , musíte do monitoru spouštěčů dodat pouze parametry *StrucId*, *Version*, *QName*a **QMgrName** , protože neodkazují na žádná jiná pole.

Zprávy se čtou z inicializační fronty a používají se ke spuštění transakcí CICS s použitím EXEC CICS START za předpokladu, že hodnota APPL_TYPE ve zprávě spouštěče je MQAT_CICS. Čtení zpráv z inicializační fronty se provádí pod řízením synchronizačního bodu CICS .

Zprávy jsou generovány, když se monitor spustí a zastaví, a když se vyskytne chyba. Tyto zprávy jsou odeslány do přechodné datové fronty CSMT.

Zde jsou dostupné verze monitoru spouštěčů:

Verze	**Použití**
amqltmc0	TXSeries 5.1 pro systémy AIX, HP-UX, Linux x86-64 a Oracle Solaris
amqltmc4	položky TXSeries 5.1 pro Windows
amqltmcc	Verze vazby klienta monitoru spouštěčů CICS

Pokud potřebujete monitor spouštěčů pro jiná prostředí, napište program, který může zpracovávat zprávy spouštěče, které správce front vloží do inicializačních front. Takový program by měl provádět následující akce:

1. Chcete-li čekat na příchod zprávy do inicializační fronty, použijte volání MQGET.
2. Prověřte pole struktury MQTM zprávy spouštěče, abyste nalezli název aplikace, která se má spustit, a prostředí, ve kterém se spustí.
3. Zadejte spouštěcí příkaz specifický pro prostředí. z/OS Například v dávkovém zpracování z/OS odešlete úlohu do interního čtecího zařízení.
4. V případě potřeby převeďte strukturu MQTM na strukturu MQTMC2 .
5. Předejte strukturu MQTMC2 nebo MQTM do spuštěné aplikace. Tato akce může obsahovat uživatelská data.
6. Přidružte k frontě aplikací aplikaci, která má sloužit k obsluze této fronty. To provedete tak, že pojmenujete objekt definice procesu (je-li vytvořen) v atributu **ProcessName** ve frontě.

   IBM i Použijte atribut DEFINE QLOCAL nebo ALTER QLOCAL. V systému IBM imůžete také použít CRTMQMQ nebo CHGMQMQ.

Další informace o rozhraní monitoru spouštěčů najdete v tématu MQTMC2.

IBM i *Monitory spouštěčů v systému IBM i*

V systému IBM inamísto řídicího příkazu **runmqtrm** použijte CL příkaz IBM MQ for IBM i **STRMQMTRM**.

Použijte příkaz STRMQMTRM následujícím způsobem:

```
STRMQMTRM INITQNAME(InitQ) MQMNAME(QMgrName)
```

Podrobnosti jsou uvedeny pro runmqtrm.

K dispozici jsou také následující vzorové programy, které můžete použít jako modely pro psaní svých vlastních monitorů spouštěčů:

AMQSTRG4
Jedná se o monitor spouštěčů, který odesílá úlohu IBM i pro proces, který má být spuštěn, ale to znamená, že existuje další zpracování přidružené ke každé zprávě spouštěče.

AMQSERV4
Toto je spouštěcí server. Pro každou zprávu spouštěče tento server spustí příkaz pro proces ve své vlastní úloze a může volat transakce CICS .

Monitor spouštěčů i spouštěcí server předávají strukturu MQTMC2 do programů, které spouštějí. Popis této struktury najdete v tématu MQTMC2. Oba tyto ukázky jsou dodávány ve zdrojovém i spustitelném tvaru.

Vzhledem k tomu, že tyto monitory spouštěčů mohou vyvolat pouze nativní programy IBM i , nemohou spouštět programy Java přímo, protože třídy Java jsou umístěny v IFS. Programy produktu Java však mohou být spouštěny nepřímo spuštěním programu v jazyce CL, který pak vyvolá program Java a prochází přes strukturu TMC2 . Minimální velikost struktury TMC2 je 732 bajtů.

Zde je uveden zdroj ukázkového příkazového procesoru (CLP):

```
PGM PARM(&TMC2)
  DCL &TMC2 *CHAR LEN(800)
    ADDENVVAR ENVVAR(TM) VALUE(&TMC2)
    QSH  CMD('java_pgmname $TM')
    RMVENVVAR ENVVAR(TM)
ENDPGM
```

Následující program monitoru spouštěčů je k dispozici pro IBM MQ MQI client: RUNMQTMC

Volejte RUNMQTMC následujícím způsobem:

```
CALL PGM(QMQM/RUNMQTMC) PARM('-m' QMgrName '-q' InitQ)
```

Vlastnosti zpráv spouštěče

V následujících tématech jsou popsány některé další vlastnosti zpráv spouštěče.

- “Perzistence a priorita zpráv spouštěče” na stránce 845
- “Restart zpráv správce front a spouštěcí zprávy” na stránce 846
- “Spouštět zprávy a změny atributů objektu” na stránce 846
- “Formát zpráv spouštěče” na stránce 846

Perzistence a priorita zpráv spouštěče

Zprávy spouštěče nejsou trvalé, protože pro ně není určen žádný požadavek.

Avšak podmínky pro generování spouštěcích událostí trvají, takže zprávy spouštěče jsou generovány, kdykoli jsou tyto podmínky splněny. Dojde-li ke ztrátě zprávy spouštěče, bude pokračovat existence zprávy aplikace ve frontě aplikací a zaručuje, že správce front vygeneruje zprávu spouštěče ihned po splnění všech podmínek.

Je-li jednotka práce odvolána, všechny zprávy spouštěče, které vygenerovala, se vždy doručí.

Zprávy spouštěče mají výchozí prioritu inicializační fronty.

Restart zpráv správce front a spouštěcí zprávy

Po restartu správce front, když je inicializační fronta otevřena pro vstup, může být do této inicializační fronty vložena zpráva spouštěče, pokud je fronta aplikací přidružená k této frontě zpráv a je definována pro spuštění.

Spouštět zprávy a změny atributů objektu

Spouštěcí zprávy se vytvářejí podle hodnot atributů spouštěče platných v době události triggeru.

Pokud není zpráva spouštěče k dispozici pro monitor spouštěčů do pozdější doby (protože zpráva, která způsobila, že byla generována, byla vložena do pracovní jednotky), žádné změny atributů triggeru mezitím nemají žádný vliv na zprávu spouštěče. Zejména zakázání spouštění nezabrání zpřístupnění zprávy spouštěče po jeho vytvoření. Také fronta aplikací nemusí již existovat v době zpřístupnění zprávy spouštěče.

Formát zpráv spouštěče

Formát zprávy spouštěče je definován strukturou MQTM.

To má následující pole, která správce front vyplní, když vytvoří zprávu spouštěče, pomocí informací v definicích objektů ve frontě aplikace a procesu přidruženého k této frontě:

StrucId
Identifikátor struktury.

Version
Verze struktury.

QName
Název fronty aplikací, na které došlo k události spouštěče. Když správce front vytvoří zprávu spouštěče, zaplní toto pole pomocí atributu **QName** ve frontě aplikací.

ProcessName
Název objektu definice procesu, který je přidružen k frontě aplikací. Když správce front vytvoří zprávu spouštěče, zaplní toto pole pomocí atributu **ProcessName** ve frontě aplikací.

TriggerData
Pole ve volném formátu pro použití monitorem spouštěčů. Když správce front vytvoří zprávu spouštěče, zaplní toto pole pomocí atributu **TriggerData** ve frontě aplikací. Na libovolném produktu IBM MQ s výjimkou IBM MQ for z/OSlze toto pole použít k uvedení názvu kanálu, který má být spuštěn.

ApplType
Typ aplikace, kterou má monitor spouštěčů spustit. Když správce front vytvoří zprávu spouštěče, vyplní toto pole pomocí atributu **ApplType** v objektu definice procesu identifikovaném v souboru *ProcessName*.

ApplId
Znakový řetězec, který identifikuje aplikaci, kterou má spustit monitor spouštěčů. Když správce front vytvoří zprávu spouštěče, vyplní toto pole pomocí atributu **ApplId** v objektu definice procesu identifikovaném v souboru *ProcessName*.

Použijete-li monitor spouštěčů CKTI dodaný produktem CICS, je atributem **ApplId** objektu definice procesu identifikátor transakce CICS .

Když používáte CSQQTRMN dodanou IBM MQ for z/OS, je atribut **ApplId** objektu definice procesu identifikátorem transakce IMS .

EnvData
Znakové pole obsahující data související s prostředním prostředím pro použití monitorem spouštěčů. Když správce front vytvoří zprávu spouštěče, vyplní toto pole pomocí atributu **EnvData** v objektu definice procesu identifikovaném v souboru *ProcessName*. The CICS-supplied trigger monitor (CKTI)

or the IBM MQ for z/OS-supplied trigger monitor (CSQQTRMN) do not use this field, but other trigger monitors might choose to use it.

UserData
Znakové pole obsahující uživatelská data pro použití monitorem spouštěčů. Když správce front vytvoří zprávu spouštěče, vyplní toto pole pomocí atributu **UserData** v objektu definice procesu identifikovaném v souboru *ProcessName*. Toto pole lze použít k určení názvu kanálu, který má být spuštěn.

V produktu MQTMje uveden úplný popis struktury zprávy spouštěče.

Když spouštění nefunguje

Program se nespustí, pokud monitor spouštěčů nemůže spustit program nebo správce front nemůže doručit zprávu spouštěče. Například ID aplikátoru v objektu procesu musí určovat, že program má být spuštěn na pozadí; jinak monitor spouštěčů nemůže spustit program.

Pokud je vytvořena zpráva spouštěče, ale nelze ji umístit do inicializační fronty (například, protože je plná fronta nebo je délka zprávy spouštěče větší než maximální délka zprávy uvedená pro inicializační frontu), spustí se zpráva spouštěče místo na frontu nedoručených zpráv (nedoručená zpráva).

Pokud operace vložení do fronty nedoručených zpráv nemůže být úspěšně dokončena, je zpráva spouštěče vyřazena a odešle se varovná zpráva z/OS do konzoly z/OS nebo do systémového operátora nebo je vložena do protokolu chyb.

Vložení zprávy spouštěče do fronty nedoručených zpráv může generovat zprávu spouštěče pro danou frontu. Tato druhá zpráva spouštěče je vyřazena, pokud přidá zprávu do fronty nedoručených zpráv.

Je-li program úspěšně spuštěn, ale ukončí se před tím, než přijme zprávu z fronty, použijte obslužný program pro trasování (například CICS AUXTRACE, pokud je program spuštěn pod CICS) a najděte příčinu selhání.

Práce s MQI a klastry

Existují speciální volby na voláních a návratových kódech, které se vztahují ke klastrování.

Použijte následující odkazy, chcete-li zjistit více o možnostech, které jsou k dispozici na voláních a návratových kódech pro použití s klastry:

- “MQOPEN a klastry” na stránce 848
- “MQPUT, MQPUT1 a klastry” na stránce 849
- “MQINQ a klastry” na stránce 849
- “MQSET a klastry” na stránce 850
- “Návratové kódy” na stránce 850

Související pojmy

“Přehled rozhraní fronty zpráv” na stránce 690
Zde jsou uvedeny informace o komponentách rozhraní MQI (Message Queue Interface).

“Připojování k správci front a odpojování od něj” na stránce 704
Chcete-li používat programovací služby IBM MQ , musí mít program připojení ke správci front. V této části se dozvíte, jak se připojit k správci front a odpojit je od správce front.

“Otevírání a zavírání objektů” na stránce 712
Tyto informace poskytují vhled do otevírání a zavírání objektů IBM MQ .

“Vložení zpráv do fronty” na stránce 723
V této části se dozvíte, jak vložit zprávy do fronty.

“Získávání zpráv z fronty” na stránce 737
Tyto informace použijte k získání informací o získávání zpráv z fronty.

“Inquaning about a nastavení atributů objektu” na stránce 815
Atributy jsou vlastnosti, které definují charakteristiky objektu IBM MQ .

“Potvrzení a zálohování jednotek práce” na stránce 818
Tyto informace popisují, jak potvrdit a zazálohovat všechny zotavitelné operace get a put, které se vyskytly v jednotce práce.

“Spuštění aplikací produktu IBM MQ pomocí spouštěčů” na stránce 829
Informace o spouštěčích a o tom, jak spustit aplikace IBM MQ pomocí spouštěčů.

“Použití a zápis aplikací v systému IBM MQ for z/OS” na stránce 851
Aplikace produktu IBM MQ for z/OS mohou být vytvořeny z programů spuštěných v mnoha různých prostředích. To znamená, že mohou využít výhody zařízení dostupných ve více než jednom prostředí.

“Aplikace mostu IMS a IMS v systému IBM MQ for z/OS” na stránce 57
Tyto informace vám pomohou při psaní aplikací produktu IMS pomocí produktu IBM MQ.

MQOPEN a klastry
Fronta, do které je při otevření fronty klastru vložena zpráva nebo z ní načtena, závisí na volání MQOPEN .

Výběr cílové fronty

Pokud nezadáte název správce front v deskriptoru objektu, produkt MQOD, správce front vybere správce front, do kterého má odeslat zprávu. Pokud v deskriptoru objektu zadáte název správce front, budou zprávy vždy odeslány do vybraného správce front.

Pokud správce front vybírá cílového správce front, závisí výběr na volbách vázání, MQOO_BIND_* a pokud existuje lokální fronta. Pokud se jedná o lokální instanci fronty, je vždy otevřená jako předvolba pro vzdálenou instanci, pokud není atribut CLWLUSEQ nastaven na hodnotu ANY. V opačném případě závisí výběr na volbách vazby. Musí být zadán buď MQOO_BIND_ON_OPEN nebo MQOO_BIND_ON_GROUP , když používáte skupiny zpráv s klastry, aby se zajistilo, že všechny zprávy ve skupině se zpracují ve stejném cíli.

Pokud správce front vybírá cílového správce front, provádí to způsobem round-robin s použitím algoritmu správy pracovní zátěže; viz Vyrovnávání pracovní zátěže v klastrech.

Je-li použit algoritmus vyrovnávání pracovní zátěže, závisí na způsobu otevření fronty klastru:

- MQOO_BIND_ON_OPEN -algoritmus se používá jednou při otevření fronty aplikací.
- MQOO_BIND_NOT_FIXED -algoritmus se používá pro každou zprávu vkládanou do fronty.
- MQOO_BIND_ON_GROUP -algoritmus se používá jednou na začátku každé skupiny zpráv.

MQOO_BIND_ON_OPEN
Volba MQOO_BIND_ON_OPEN na volání MQOPEN určuje, že má být správce cílové fronty opraven. Použijte volbu MQOO_BIND_ON_OPEN , pokud existuje více instancí stejné fronty v rámci klastru. Všechny zprávy vkládané do fronty s určením manipulátoru objektu vráceného z volání produktu MQOPEN jsou směrovány do stejného správce front.

- Pokud mají zprávy afinity, použijte volbu MQOO_BIND_ON_OPEN . Je-li například celá dávka zpráv zpracovávána ve stejném správci front, zadejte při otevření fronty produkt MQOO_BIND_ON_OPEN . Produkt IBM MQ opravuje správce front a přenosovou cestu, která má být přijata všemi zprávami vkládané do této fronty.
- Je-li zadána volba MQOO_BIND_ON_OPEN , musí být fronta znovu otevřena pro novou instanci fronty, která má být vybrána.

MQOO_BIND_NOT_FIXED
Volba MQOO_BIND_NOT_FIXED ve volání MQOPEN určuje, že cílový správce front není pevný. Zprávy zapsané do fronty s uvedením ovladače objektu vrácené z volání MQOPEN jsou směrovány do správce front v MQPUT čase na základě zpráv. Volbu MQOO_BIND_NOT_FIXED použijte, pokud nechcete, aby všechny vaše zprávy byly zapsány do stejného cíle.

- Nezadávejte MQOO_BIND_NOT_FIXED a MQMF_SEGMENTATION_ALLOWED zároveň. Pokud tak učiníte, segmenty vaší zprávy mohou být dodány různým správcům front, rozptýleným v celém klastru.

MQOO_BIND_ON_GROUP
Umožňuje aplikaci požadovat, aby byla skupina zpráv alokována do stejné cílové instance. Tato volba je platná pouze pro fronty a má vliv pouze na fronty klastru. Je-li tato volba zadána pro frontu, která není frontou klastru, je tato volba ignorována.

- Skupiny jsou směrovány pouze do jednoho místa určení, je-li MQPMO_LOGICAL_ORDER na příkazu MQPUT určeno. Je-li zadán parametr MQOO_BIND_ON_GROUP, ale zpráva není součástí logické skupiny, použije se místo něj hodnota BIND_NOT_FIXED.

MQOO_BIND_AS_Q_DEF
Pokud nezadáte ani MQOO_BIND_ON_OPEN, MQOO_BIND_NOT_FIXED nebo MQOO_BIND_ON_GROUP, standardní volba je MQOO_BIND_AS_Q_DEF. Použití MQOO_BIND_AS_Q_DEF způsobí, že vazba, která se použije pro popisovač fronty, bude převzata z atributu fronty DefBind .

Relevance voleb obslužného programu MQOPEN

Volby MQOPEN MQOO_BROWSE, MQOO_INPUT_*nebo MQOO_SET vyžadují, aby byla lokální instance fronty klastru MQOPEN úspěšná.

Volby MQOPEN , MQOO_OUTPUT, MQOO_BIND_*nebo MQOO_INQUIRE nevyžadují lokální instanci fronty klastru, aby uspěly.

Vyřešený název správce front

Když je název správce front vyřešen v MQOPEN čase, je vyřešený název vrácen do aplikace. Pokud se aplikace pokusí použít tento název při dalším volání produktu MQOPEN , může zjistit, že nemá autorizaci pro přístup k názvu.

MQPUT, MQPUT1 a klastry

Je-li MQOO_BIND_NOT_FIXED zadán na MQOPEN , rutiny správy pracovní zátěže zvolí, které místo určení MQPUT nebo MQPUT1 vyberte.

Je-li MQOO_BIND_NOT_FIXED zadán ve volání příkazu MQOPEN , každé následující volání příkazu MQPUT vyvolá rutinu správy pracovní zátěže a určí správce front, kterému má být zpráva odeslána. Místo určení a cesta, které mají být vybrány, jsou vybrány na základě zpráv po zprávě. Místo určení a cesta se mohou po vložení zprávy změnit, pokud se změní podmínky v síti. Volání MQPUT1 vždy funguje, jako kdyby MQOO_BIND_NOT_FIXED byl v platnosti, tj. vždy vyvolává rutinu správy pracovní zátěže.

Když rutina správy pracovní zátěže vybrala správce front, dokončí operaci vložení lokální správce front. Zprávu lze umístit do jiných front:

1. Je-li cílem lokální instance fronty, zpráva se umístí do lokální fronty.
2. Je-li cílem správce front v klastru, bude zpráva vložena do přenosové fronty klastru.
3. Je-li cílem správce front mimo klastr, bude zpráva vložena do přenosové fronty se stejným názvem jako má cílový správce front.

Je-li v volání MQOPEN zadán parametr MQOO_BIND_ON_OPEN , volání MQPUT nespouští rutinu správy pracovní zátěže, protože cíl a trasa již byly vybrány.

MQINQ a klastry

Která fronta klastru je inquired upon závisí na volbách, které kombinujete s MQOO_INQUIRE.

Než se můžete dotázat na frontu, otevřete ji pomocí volání MQOPEN a zadejte MQOO_INQUIRE.

Chcete-li se dotázat na frontu klastru, použijte volání MQOPEN a zkombinujte ostatní volby s MQOO_INQUIRE. Atributy, které mohou být dotazovány, závisí na tom, zda existuje lokální instance fronty klastru a jak je fronta otevřena:

- Kombinace MQOO_BROWSE, MQOO_INPUT_*nebo MQOO_SET s MQOO_INQUIRE vyžaduje lokální instanci fronty klastru, aby byla otevřená úspěšná. V tomto případě se můžete dotázat na všechny atributy, které jsou platné pro lokální fronty.

- Kombinace MQOO_OUTPUT s MQOO_INQUIREa uvedení žádné z předchozích voleb, instance otevřené je buď:
  - Instance na lokálním správci front, pokud existuje. V tomto případě se můžete dotázat na všechny atributy, které jsou platné pro lokální fronty.
  - Instance na jiném místě v klastru, pokud neexistuje žádná lokální instance správce front. V tomto případě mohou být dotazovány pouze následující atributy. Atribut QType má hodnotu MQQT_CLUSTER v tomto případě.
    - DefBind
    - DefPersistence
    - DefPriority
    - InhibitPut
    - QDesc
    - QName
    - QTYPE

Chcete-li se dotázat na atribut DefBind ve frontě klastru, použijte volání MQINQ s voličem MQIA_DEF_BIND. Vrácená hodnota je buď MQBND_BIND_ON_OPEN , nebo MQBND_BIND_NOT_FIXED, nebo MQBND_BIND_ON_GROUP. Při použití skupin s klastry musí být zadán buď MQBND_BIND_ON_OPEN nebo MQBND_BIND_ON_GROUP .

Chcete-li se dotázat na atributy CLUSTER a CLUSNL na lokální instanci fronty, použijte volání MQINQ s voličem MQCA_CLUSTER_NAME nebo voličem MQCA_CLUSTER_NAMELIST.

Poznámka: Pokud otevřete frontu klastru bez určení fronty, ke které je operace MQOPEN svázána, může se následující volání produktu MQINQ dotazovat na různé instance této fronty klastru.

Související pojmy

“Volba MQOPEN pro frontu klastru” na stránce 719

Vazba použitá pro popisovač fronty je převzata z atributu fronty produktu **DefBind** , který může převzít hodnotu MQBND_BIND_ON_OPEN, MQBND_BIND_NOT_FIXEDnebo MQBND_BIND_ON_GROUP.

MQSET a klastry

The MQOPEN option MQOO_SET option requires there to be a local instance of a cluster queue for MQSET to succeed.

Volání MQSET nelze použít k nastavení atributů fronty jinde v klastru.

Můžete otevřít lokální alias nebo vzdálenou frontu definovanou s atributem klastru a použít volání MQSET . Atributy lokálního aliasu nebo vzdálené fronty můžete nastavit. Pokud je cílová fronta frontou klastru definovanou v jiném správci front, nezáleží na tom, zda je cílová fronta definována.

Návratové kódy

Návratové kódy specifické pro klastry

MQRC_CLUSTER_EXIT_ERROR (2266 X'8DA')

Volání MQOPEN, MQPUTnebo MQPUT1 se vydává za účelem otevření fronty klastru nebo vložení zprávy do fronty. Uživatelská procedura pracovní zátěže klastru, definovaná atributem ClusterWorkloadExit správce front, neočekávaně selže nebo neodpovídá času.

Zpráva se zapíše do systémového protokolu na IBM MQ for z/OS , kde získáte více informací o této chybě.

Následné volání MQOPEN, MQPUTa MQPUT1 pro tento popisovač fronty se zpracují, jako by byl atribut ClusterWorkloadExit prázdný.

MQRC_CLUSTER_EXIT_LOAD_ERROR (2267 X'8DB')

V systémech z/OSnelze uživatelskou proceduru pracovní zátěže klastru načíst.

Zpráva je zapsána do systémového protokolu a zpracování pokračuje, jako by atribut `ClusterWorkloadExit` byl prázdný.

Multi V systému Multiplatformyse pro připojení ke správci front vydá volání `MQCONN` nebo `MQCONNX` . Volání se nezdaří, protože nelze načíst uživatelskou proceduru pracovní zátěže klastru, která je definována atributem správce front `ClusterWorkloadExit` správce front.

MQRC_CLUSTER_PUT_INHIBITED (2268 X'8DC')

Pro frontu klastru je vydáno volání `MQOPEN` s volbami `MQOO_OUTPUT` a `MQOO_BIND_ON_OPEN` v platnosti. Všechny instance fronty v klastru jsou momentálně blokovány tím, že má atribut `InhibitPut` nastaven na hodnotu `MQQA_PUT_INHIBITED`. Vzhledem k tomu, že nejsou k dispozici žádné instance fronty pro příjem zpráv, volání `MQOPEN` se nezdaří.

Tento kód příčiny se objevuje pouze v případě, že jsou splněny obě následující podmínky:

- Neexistuje žádná lokální instance fronty. Existuje-li lokální instance, volání `MQOPEN` bude úspěšné i v případě, že lokální instance bude blokována.
- Pro frontu neexistuje žádná uživatelská procedura pracovní zátěže klastru, nebo je zde uživatelská procedura pracovní zátěže klastru, ale nevybírá instanci fronty. (Pokud uživatelská procedura pracovní zátěže klastru zvolí instanci fronty, bude volání `MQOPEN` úspěšné, a to i v případě, že je tato instance zakázána.)

Je-li u volání `MQOPEN` zadána volba `MQOO_BIND_NOT_FIXED` , může být volání úspěšné i v případě, že všechny fronty v klastru budou mít blokováno vkládání. Následující volání příkazu `MQPUT` se však může nezdařit, pokud jsou všechny fronty stále blokovány v době volání.

MQRC_CLUSTER_RESOLUTION_ERROR (2189 X'88D')

1. Volání `MQOPEN`, `MQPUT`nebo `MQPUT1` se vydává za účelem otevření fronty klastru nebo vložení zprávy do fronty. Definici fronty nelze správně rozpoznat, protože je vyžadována odezva od správce front úplného úložiště, ale žádná není k dispozici.
2. Volání `MQOPEN`, `MQPUT`, `MQPUT1` nebo `MQSUB` je vydáno pro objekt tématu s určením `PUBSCOPE ( ALL)` nebo `SUBSCOPE ( ALL)`. Definici tématu klastru nelze správně rozpoznat, protože je vyžadována odezva od správce front úplného úložiště, ale žádná není k dispozici.

MQRC_CLUSTER_RESOURCE_ERROR (2269 X'8DD')

Volání `MQOPEN`, `MQPUT`nebo `MQPUT1` je vydáno pro frontu klastru. Vyskytuje se chyba při pokusu o použití prostředku požadovaného pro klastrování.

MQRC_NO_DESTINATIONS_AVAILABLE (2270 X'8DE')

Bylo zadáno volání `MQPUT` nebo `MQPUT1` , které má vložit zprávu do fronty klastru. V době volání již v klastru nejsou žádné instance fronty. `MQPUT` selže a zpráva se neodešle.

Chyba se může vyskytnout, pokud je `MQOO_BIND_NOT_FIXED` zadán ve volání `MQOPEN` , který otevírá frontu, nebo `MQPUT1` se používá k vložení zprávy.

MQRC_STOPPED_BY_CLUSTER_EXIT (2188 X'88C')

Volání `MQOPEN`, `MQPUT`nebo `MQPUT1` se vydává za účelem otevření nebo vložení zprávy do fronty klastru. Uživatelská procedura pracovní zátěže klastru odmítne volání.

z/OS Použití a zápis aplikací v systému IBM MQ for z/OS

Aplikace produktu IBM MQ for z/OS mohou být vytvořeny z programů spuštěných v mnoha různých prostředích. To znamená, že mohou využít výhody zařízení dostupných ve více než jednom prostředí.

Tyto informace vysvětlují zařízení produktu IBM MQ dostupná pro programy spuštěné v každém z podporovaných prostředí. Kromě toho,

- Informace o použití aplikace IBM MQ-CICS bridgenaleznete v tématu Použití produktu IBM MQ s produktem CICS .

- Informace o použití produktu IMS a mostu IMS naleznete v tématu “Aplikace mostu IMS a IMS v systému IBM MQ for z/OS” na stránce 57.

Následující odkazy použijte k vyhledání dalších informací o používání a zápisu aplikací na serveru IBM MQ for z/OS:

- “Funkce produktu IBM MQ for z/OS závislé na prostředí” na stránce 852
- “Debugovací zařízení, podpora synchronizačního bodu a podpora zotavení” na stránce 853
- “Rozhraní produktu IBM MQ for z/OS s prostředím aplikace” na stránce 854
- “Zápis aplikací produktu z/OS UNIX System Services” na stránce 855
- “Programování aplikací se sdílenými frontami” na stránce 859

Související pojmy

“Přehled rozhraní fronty zpráv” na stránce 690
Zde jsou uvedeny informace o komponentách rozhraní MQI (Message Queue Interface).

“Připojování k správci front a odpojování od něj” na stránce 704
Chcete-li používat programovací služby IBM MQ , musí mít program připojení ke správci front. V této části se dozvíte, jak se připojit k správci front a odpojit je od správce front.

“Otevírání a zavírání objektů” na stránce 712
Tyto informace poskytují vhled do otevírání a zavírání objektů IBM MQ .

“Vložení zpráv do fronty” na stránce 723
V této části se dozvíte, jak vložit zprávy do fronty.

“Získávání zpráv z fronty” na stránce 737
Tyto informace použijte k získání informací o získávání zpráv z fronty.

“Inquaning about a nastavení atributů objektu” na stránce 815
Atributy jsou vlastnosti, které definují charakteristiky objektu IBM MQ .

“Potvrzení a zálohování jednotek práce” na stránce 818
Tyto informace popisují, jak potvrdit a zazálohovat všechny zotavitelné operace get a put, které se vyskytly v jednotce práce.

“Spuštění aplikací produktu IBM MQ pomocí spouštěčů” na stránce 829
Informace o spouštěčích a o tom, jak spustit aplikace IBM MQ pomocí spouštěčů.

“Práce s MQI a klastry” na stránce 847
Existují speciální volby na voláních a návratových kódech, které se vztahují ke klastrování.

“Aplikace mostu IMS a IMS v systému IBM MQ for z/OS” na stránce 57
Tyto informace vám pomohou při psaní aplikací produktu IMS pomocí produktu IBM MQ.

Funkce produktu IBM MQ for z/OS závislé na prostředí

Tyto informace použijte při zvažování funkcí produktu IBM MQ for z/OS .

Hlavní rozdíly, které je třeba vzít v úvahu mezi funkcemi produktu IBM MQ v prostředích, ve kterých IBM MQ for z/OS běží, jsou:

- IBM MQ for z/OS poskytuje následující monitory spouštěčů:
  - CKTI pro použití v prostředí CICS
  - CSQQTRMN pro použití v prostředí IMS

  Chcete-li spouštět aplikace v jiných prostředích, musíte napsat svůj vlastní modul.
- V prostředí CICS a IMS je podporováno použití dvoufázového potvrzování. Je také podporován v dávkovém prostředí produktu z/OS pomocí správy transakcí a zotavitelných služeb správce prostředků (RRS). Jednofázové potvrzování je podporováno v prostředí z/OS samotným IBM MQ .
- V případě dávkových a IMS prostředí poskytuje rozhraní MQI volání pro připojení programů a jejich odpojení od správce front. Programy se mohou připojovat k více než jednomu správci front.

- Systém CICS se může připojit pouze k jednomu správci front. K tomu může dojít, když je iniciováno CICS , pokud je název subsystému definován ve spouštěcí úloze systému CICS . Volání připojení MQI a odpojení jsou tolerovány, ale nemají žádný účinek, v prostředí produktu CICS .
- Ukončení přejezdu rozhraní API umožňuje programu zasahovat do zpracování všech volání MQI. Tato uživatelská procedura je k dispozici pouze v prostředí produktu CICS .
- V produktu CICS v systémech s více procesory je získána některá výkonnostní výhoda, protože volání MQI lze provést pod více z/OS TCB. Další informace naleznete v příručce Plánování v systému z/OS *IBM MQ for z/OS Concepts and Planning Guide*.

Tyto funkce jsou shrnuty v Tabulka 115 na stránce 853.

Tabulka 115. z/OS funkce prostředí

	CICS	**IMS**	**Štek/TSO**
Je dodán monitor spouštěčů	Ano	Ano	Ne
Dvoufázové potvrzení	Ano	Ano	Ano
jednofázové potvrzení	Ano	Ne	Ano
Připojit/odpojit volání MQI	Tolerovaný	Ano	Ano
Uživatelská procedura napříč rozhraním API	Ano	Ne	Ne

Poznámka: Dvoufázové potvrzování je podporováno v prostředí Batch/TSO pomocí RRS.

Debugovací zařízení, podpora synchronizačního bodu a podpora zotavení

Tyto informace použijte, chcete-li se dozvědět více o zařízeních pro ladění programu, podpoře synchronizačního bodu a podpoře pro obnovu.

Zařízení pro ladění programu

Produkt IBM MQ for z/OS poskytuje trasovací prostředek, který můžete použít k ladění vašich programů ve všech prostředích.

Navíc v prostředí produktu CICS můžete použít:

- Diagnostický prostředek provedení CICS (CEDF)
- Řídicí transakce trasování CICS (CETR)
- Uživatelská procedura přejezdu rozhraní API produktu IBM MQ for z/OS

Na platformě z/OS můžete použít jakýkoli dostupný interaktivní ladicí nástroj podporovaný programovacím jazykem, který používáte.

Podpora synchronizačních bodů

Synchronizace začátku a konce jednotek práce je nezbytná v prostředí zpracování transakcí, aby bylo možné bezpečně používat zpracování transakcí.

To je plně podporováno produktem IBM MQ for z/OS v prostředích CICS a IMS . Plná podpora znamená spolupráci mezi správci prostředků, aby jednotky práce mohly být potvrzeny nebo zálohovány v uniformě, pod kontrolou CICS nebo IMS. Příklady správců prostředků jsou Db2, CICS Řízení souboru, IMSa IBM MQ for z/OS.

Dávkové aplikace produktu z/OS mohou používat volání produktu IBM MQ for z/OS k poskytnutí prostředku jednofázového potvrzování. To znamená, že sada operací fronty definované aplikací může být potvrzena nebo vrácena bez odkazu na jiné správce prostředků.

Dvoufázové potvrzování je podporováno také v dávkovém prostředí produktu z/OS pomocí správy transakcí a zotavitelných služeb správce prostředků (RRS). Další informace viz téma Synchronizační body v dávkových aplikacích produktu z/OS.

Podpora obnovy

Je-li připojení mezi správcem front a systémem CICS nebo IMS během transakce přerušeno, některé jednotky práce nemusí být úspěšně zálohovány.

Tyto jednotky práce jsou však vyřešeny správcem front (pod kontrolou správce synchronizačního bodu), jakmile se znovu ustanoví jeho připojení k systému CICS nebo IMS .

Rozhraní produktu IBM MQ for z/OS s prostředím aplikace

Chcete-li povolit, aby aplikace spuštěné v různých prostředích odesílaly a přijímaly zprávy prostřednictvím sítě front zpráv, poskytuje produkt IBM MQ for z/OS *adaptér* pro každé prostředí, které podporuje.

Tyto adaptéry jsou rozhraní mezi aplikačními programy a subsystémy IBM MQ for z/OS . Umožňují programům používat rozhraní MQI.

Dávkový adaptér

Tyto informace použijte, chcete-li se dozvědět více o adaptéru pro dávkové zpracování a o potvrzovacího protokolu, který podporuje.

Dávkový adaptér poskytuje přístup k prostředkům produktu IBM MQ for z/OS pro programy spuštěné v:

- Režim úlohy (TCB)
- Stav problému nebo supervizora
- Řídicí režim primárního adresního prostoru

Programy nesmí být v režimu cross-memory.

Spojení mezi aplikačními programy a IBM MQ for z/OS jsou na úrovni úlohy. Adaptér poskytuje jeden podproces připojení z řídicího bloku úloh aplikace (TCB) do produktu IBM MQ for z/OS.

Adaptér podporuje protokol jednofázového potvrzování pro změny provedené u prostředků vlastněných produktem IBM MQ for z/OS ; nepodporuje protokoly multiphase-commit.

Dávkový adaptér RRS

Tyto informace použijte k získání informací o adaptéru dávky RRS a dvou dávkových adaptérech RRS, které poskytuje produkt IBM MQ.

Správa transakcí a zotavitelné služby správce prostředků (RRS):

- Používá z/OS RRS pro vázané zpracování.
- Podporuje simultánní připojení k více subsystémům IBM MQ spuštěným na jediné instanci produktu z/OS z jediné úlohy.
- Provides z/OS-wide coordinated commitment control (using z/OS RRS) for recoverable resources accessed through z/OS RRS-compliant recoverable managers for:
  - Aplikace, které se připojují k produktu IBM MQ pomocí dávkového adaptéru RRS.
  - Db2-stored procedures executing in a Db2-stored procedures address space that is managed by a workload manager (WLM) on z/OS.
- Podporuje schopnost přepnout dávkové vlákno IBM MQ mezi TCB.

Produkt IBM MQ for z/OS poskytuje dva dávkové adaptéry RRS:

CSQBRSTB
Tento adaptér vyžaduje, abyste změnili libovolný příkaz MQCMIT na SRRCMIT a všechny příkazy MQBACK pro SRRBACK ve vaší aplikaci IBM MQ . (Pokud kód MQCMIT nebo MQBACK v aplikaci propojené s CSQBRSTB, obdržíte ERROR (MQRC_ENVIRONMENT_ERROR).)

CSQBRRSI
Tento adaptér umožňuje aplikaci IBM MQ použít buď MQCMIT a MQBACK, nebo SRRCMIT a SRRBACK.

Poznámka: CSQBRSTB a CSQBRRSI se dodávají s atributy sestavení AMODE (31) RMODE (ANY). Pokud vaše aplikace načte buď kód stub pod hranicí 16 MB, nejprve znovu propojte stub s RMODE (24).

Migration

Existující aplikace Batch/TSO IBM MQ můžete migrovat, chcete-li používat koordinaci služby RRS s několika nebo žádnými změnami.

Pokud odkazujete na aplikaci IBM MQ s adaptérem CSQBRRSI, MQCMIT a MQBACK syncpoint, bude vaše jednotka pracovat napříč produktem IBM MQ a všemi dalšími správci prostředků s podporou RRS. Pokud odkazujete-upravíte aplikaci IBM MQ s adaptérem CSQBRSTB, změňte MQCMIT na SRRCMIT a MQBACK na SRRBACK. Posledně uvedený přístup je vhodnější; jasně označuje, že synchronizační bod není omezen pouze na prostředky produktu IBM MQ .

Adaptér IMS
Používáte-li adaptér IMS ze systému IBM MQ for z/OS , ujistěte se, že produkt IMS může získat dostatečné úložiště pro ukládání zpráv až 100 MB.

Poznámka pro uživatele

Adaptér *IMS* poskytuje přístup k prostředkům produktu IBM MQ for z/OS pro:

- Programy pro zpracování zpráv online (MPP)
- Interaktivní programy rychlých cest (IFP)
- Programy pro zpracování dávkových zpráv (BMP)

Chcete-li tyto prostředky použít, musí být programy spuštěny v režimu úlohy (TCB) a ve stavu problému. Nesmí být v režimu křížové paměti ani v režimu přístupu k registru.

Adaptér poskytuje podproces připojení z řídicího bloku úloh aplikace (TCB) do produktu IBM MQ. Adaptér podporuje protokol s dvoufázovým potvrzováním pro změny provedené u prostředků vlastněných produktem IBM MQ for z/OS, přičemž produkt IMS působí jako koordinátor synchronizačního bodu.

Adaptér také poskytuje program pro monitorování spouštěčů, který může spouštět programy automaticky, když jsou splněny určité podmínky spouštěče ve frontě. Další informace viz “Spuštění aplikací produktu IBM MQ pomocí spouštěčů” na stránce 829.

Pokud zapisujete dávkové programy DL/I, postupujte podle pokynů uvedených v tomto tématu pro dávkové programy z/OS .

Zápis aplikací produktu z/OS UNIX System Services

Dávkový adaptér podporuje připojení správce front z dávkových úloh a adresních prostorů TSO:

Pokud uvažujeme o adresovém prostoru dávky, adaptér podporuje připojení z více TCB v rámci tohoto adresního prostoru takto:

- Každý TCB se může připojit k více správcům front pomocí volání MQCONN nebo MQCONNX (ale TCB může mít v jednom okamžiku pouze jednu instanci připojení k určitému správci front).
- Více TCB se může připojit ke stejnému správci front (ale popisovač správce front vrácený při každém volání MQCONN nebo MQCONNX je svázán s vydáním TCB a nemůže jej použít žádná jiná TCB).

Produkt z/OS UNIX System Services podporuje dva typy volání pthread_create:

1. Nebežná vlákna, spusťte jeden pro každou TCB, která jsou PŘIPOJNÁ a DETACHed na začátku vlákna a končí z/OS.
2. Vlákna střední váhy, jeden pro každou TCB, ale TCB může být jeden z poolu long-running TCB. Aplikace musí provést všechny potřebné vyčištění aplikace, protože je-li připojení k serveru připojeno, výchozí ukončení podprocesu, které může být poskytnuto serverem při ukončení úlohy (TCB), se **ne** vždy řídí.

Odlehčené podprocesy nejsou podporovány. (Pokud aplikace vytvoří trvalé podprocesy, které odbaví své vlastní pracovní požadavky, je ***aplikace*** zodpovědná za vyčištění jakýchkoli prostředků před spuštěním dalšího pracovního požadavku.)

Produkt IBM MQ for z/OS podporuje podprocesy produktu z/OS UNIX System Services s použitím dávkového adaptéru následujícím způsobem:

1. Nebežná vlákna jsou plně podporována jako dávková spojení. Každý podproces se spustí ve své vlastní TCB, který je připojen a odpojen od začátku a konce podprocesu. Pokud má podproces skončit před vyvoláním volání MQDISC, IBM MQ for z/OS provede standardní vyčištění úlohy, které zahrnuje potvrzení všech nevyřízených pracovních jednotek, pokud je podproces normálně ukončen, nebo zálohování, pokud došlo k nestandardnímu ukončení podprocesu.
2. Vlákna střední váhy jsou plně podporována, ale pokud bude TCB znovu použita jiným podprocesem, musí aplikace zajistit, aby volání MQDISC, kterému předchází buď MQCMIT nebo MQBACK, bylo vydáno dříve, než se spustí další vlákno. Z toho vyplývá, že pokud aplikace vytvořila obslužnou rutinu přerušení programu a poté dojde k ukončení aplikace, musí obslužná rutina přerušení vydávat volání MQCMIT a MQDISC předtím, než bude TCB znovu použit pro jiný podproces.

Poznámka: Modely podprocesů **nepodporují** přístup ke společným prostředkům produktu IBM MQ z více podprocesů.

Uživatelská procedura přeletu rozhraní API pro z/OS

Toto téma obsahuje informace o programovacím rozhraní, které jsou citlivé na produkt.

Ukončení je bod v kódu IBM, kde můžete spustit vlastní kód. Produkt IBM MQ for z/OS poskytuje *uživatelskou proceduru křížení rozhraní API* , kterou lze použít k zachycení volání do rozhraní MQI a k monitorování nebo úpravě funkce volání MQI. Tento oddíl popisuje použití uživatelské procedury pro přechod přes rozhraní API a popisuje vzorový uživatelský program, který je dodáván s produktem IBM MQ for z/OS.

Tato sekce je použitelná pouze pro uživatele produktu CICS TS V3.1 a starší. Uživatelé produktu CICS TS V3.2 a pozdějších by měli odkazovat na sekci CICS Integrace s produktem IBM MQ v dokumentaci produktu CICS .

Poznámka

Uživatelská procedura překřížení rozhraní API je vyvolána pouze adaptérem CICS produktu IBM MQ for z/OS. Ukončovací program běží v adresním prostoru CICS .

Vytvoření vlastního ukončovacího programu
Můžete použít vzorový výstupní program přeletu rozhraní API (CSQCAPX) dodávaný s IBM MQ for z/OS jako framework pro váš vlastní program.

Tento popis je popsán v tématu “Ukázkový výstupní program přeletu rozhraní API CSQCAPX” na stránce 857.

Při zápisu ukončovacího programu k vyhledání názvu volání MQI vydaného aplikací prozkoumejte pole *ExitCommand* struktury MQXP. Chcete-li zjistit počet parametrů v rámci volání, zkontrolujte pole *ExitParmCount* . Můžete použít 16bajtové pole *ExitUserArea* k uložení adresy libovolného dynamického úložiště, které aplikace získá. Toto pole je uchováno mezi vyvoláními ukončení a má stejnou dobu trvání jako úloha CICS .

Pokud používáte CICS Transaction Server V3.2, musíte napsat svůj uživatelský program, který má zajistit neporušenost vláken, a deklarovat uživatelský program jako neporušenost vláken. Pokud používáte starší verze produktu CICS , doporučuje se také zapsat a deklarovat vaše výstupní programy jako zabezpečené podprocesy, které jsou připraveny k migraci na server CICS Transaction Server V3.2.

Ukončovací program může potlačit provedení volání MQI vrácením MQXCC_SUPPRESS_FUNCTION nebo MQXCC_SKIP_FUNCTION v poli *ExitResponse* . Chcete-li, aby volání bylo provedeno (a ukončovací program byl znovu vyvolán po dokončení volání), váš výstupní program musí vrátit MQXCC_OK.

Je-li vyvolán po volání MQI, může výstupní program zkontrolovat a upravit kódy dokončení a kódy příčiny nastavené voláním.

Poznámky k použití

Zde jsou některé obecné body, které je třeba vzít v úvahu při psaní svého výstupního programu:

- Z výkonnostních důvodů napište svůj program v assembleru-language. Pokud jej napíšete do žádného z dalších jazyků podporovaných produktem IBM MQ for z/OS, musíte zadat vlastní soubor definice dat.
- Link-edit your program as AMODE (31) and RMODE (ANY).
- Chcete-li definovat blok výstupních parametrů pro svůj program, použijte makro assembler-language, CMQXPA.
- Zadejte CONCURRENCY (THREADSAFE), když definujete svůj ukončovací program a všechny programy, které vaše výstupní program volá.
- Používáte-li funkci ochrany dat produktu CICS Transaction Server for z/OS , musí být váš program spuštěn v klíči pro provádění CICS . To znamená, že musíte uvést EXECKEY (CICS) při definování jak výstupního programu, tak všech programů, na které se tento výstupní program řídí. Informace o ukončovacích programech CICS a o zařízeních pro ochranu úložného prostoru CICS naleznete v příručce *CICS Customization Guide*.
- Váš program může používat všechna rozhraní API (například IMS, Db2a CICS). který může použít uživatelský program související s úlohou produktu CICS . Může také použít kterékoli z volání MQI s výjimkou volání MQCONN, MQCONNX a MQDISC. Avšak žádná volání MQI v rámci uživatelského programu nevyvolá program výstupního bodu podruhé.
- Váš program může vydat příkazy EXEC CICS SYNCPOINT nebo EXEC CICS SYNCPOINT ROLLBACK. Tyto příkazy však potvrdí nebo vrátí zpět ***všechny*** aktualizace provedené úlohou až do bodu, kdy byla uživatelská procedura použita, a jejich použití se proto nedoporučuje.
- Váš program musí být ukončen zadáním příkazu EXEC CICS RETURN. Nesmí přenášet ovládací prvek s příkazem XCTL.
- Uživatelské procedury jsou zapsány jako rozšíření kódu produktu IBM MQ for z/OS . Zajistěte, aby váš výstup nenarušil žádné programy nebo transakce IBM MQ for z/OS , které používají rozhraní MQI. Ty jsou obvykle označeny prefixem CSQ nebo CK.
- Je-li CSQCAPX definován pro CICS, pokusí se systém CICS načíst uživatelský program, když se CICS připojí k IBM MQ for z/OS. Je-li tento pokus úspěšný, bude odeslána zpráva CSQC301I do panelu CKQC nebo do konzoly systému. Je-li načtení neúspěšné (například, pokud zaváděcí modul neexistuje v žádné z knihoven ve zřetězení DFHRPL), odešle se zpráva CSQC315 do panelu CKQC nebo do konzoly systému.
- Protože parametry v komunikační oblasti jsou adresy, výstupní program musí být definován jako lokální vzhledem k systému CICS (to jest ne jako vzdálený program).

Ukázkový výstupní program přeletu rozhraní API CSQCAPX
Ukázkový ukončovací program se dodává jako program v assembleru. Zdrojový soubor (CSQCAPX) je dodán v knihovně **thlqual**.SCSQASMS (kde **thlqual** je vysokoúrovňový kvalifikátor používaný vaší instalací). Tento zdrojový soubor zahrnuje pseudokód, který popisuje logiku programu.

Ukázkový program obsahuje inicializační kód a rozvržení, které můžete použít při psaní svých vlastních ukončovacích programů.

Ukázka ukazuje, jak:

- Nastavení bloku výstupních parametrů
- Adresování bloků parametrů volání a ukončení
- Určit, pro které volání MQI je vyvoláno ukončení
- Určete, zda je ukončení vyvoláno před zpracováním nebo po zpracování volání MQI
- Vložit zprávu do dočasné paměťové fronty CICS
- Použití makra DFHEIENT pro uchování dynamického úložiště pro zachování opětovné entranccy

- Použít DFHEIBLK pro řídicí blok rozhraní exec CICS
- Chybové stavy depeše
- Návrat řízení volajícímu

Návrh vzorového ukončovacího programu

Ukázkový výstupní program zapisuje zprávy do dočasné paměťové fronty CICS (CSQ1EXIT), aby se zobrazila operace ukončení.

Zprávy zobrazují, zda je ukončení vyvoláno před voláním MQI nebo po něm. Je-li uživatelská procedura vyvolána po volání, zpráva obsahuje kód dokončení a kód příčiny vrácený voláním. Ukázka používá pojmenované konstanty z makra CMQXPA pro kontrolu typu položky (tj. před voláním nebo po něm).

Ukázka neprovádí žádnou funkci monitorování, ale jednoduše umisťuje časově orazítkované zprávy do fronty CICS označující typ volání, který zpracovává. To poskytuje indikaci výkonu rozhraní MQI, stejně jako správné fungování výstupního programu.

Poznámka: Ukázkový ukončovací program vydává šest EXEC CICS volání pro každé volání MQI, které bylo provedeno v době spuštění programu. Pokud použijete tento uživatelský program, výkon produktu IBM MQ for z/OS se sníží.

Příprava a použití uživatelské procedury pro přechod na rozhraní API
Ukázka uživatelské procedury je k dispozici pouze ve zdrojovém tvaru.

Chcete-li použít vzorovou uživatelskou proceduru nebo uživatelský program, který jste vytvořili, vytvořte zaváděcí knihovnu stejně jako u všech ostatních programů produktu CICS , jak je popsáno v tématu “Sestavování aplikací produktu CICS v produktu z/OS” na stránce 1004.

- V případě CICS Transaction Server pro z/OS a CICS for MVS/ESA, když aktualizujete datovou sadu definic systému CICS (CSD), definice, které potřebujete, jsou ve členu **thlqual**.SCSQPROC (CSQ4B100).

  Poznámka: Definice používají příponu MQ. Je-li tato přípona již použita ve vašem podniku, musí být tato přípona změněna před fází sestavení.

Použijete-li výchozí poskytnuté definice programu CICS , je výstupní program CSQCAPX instalován ve stavu ***disabled*** . Je tomu tak proto, že použití výstupního programu může vést k výraznému snížení výkonu.

Chcete-li dočasně aktivovat proceduru překročení rozhraní API, postupujte takto:

1. Vydejte příkaz **CEMT S PROGRAM(CSQCAPX) ENABLED** z hlavního terminálu CICS .
2. Spusťte transakci CKQC a použijte volbu 3 ve stahovací nabídce Připojení ke změně stavu uživatelské procedury rozhraní API-přechod na **Povoleno**.

Chcete-li spustit produkt IBM MQ for z/OS s trvale ukončovanou uživatelskou procedurou překřížení rozhraní API, proveďte u produktu CICS Transaction Server for z/OS a CICS for MVS/ESAjednu z následujících možností:

- Upravte definici CSQCAPX v členu CSQ4B100, změnou STATUS (DISABLED) na STATUS (ENABLED). Definici CSD CICS můžete aktualizovat pomocí obslužného programu dávkového programu DFHCSDUP dodaného s CICS.
- Změňte definici CSQCAPX ve skupině CSQCAT1 změnou stavu ZAKÁZÁN na AKTIVOVÁN.

V obou případech musíte přeinstalovat skupinu. To můžete provést za studena-spuštěním vašeho systému CICS nebo pomocí transakce CICS CEDA, chcete-li přeinstalovat skupinu, zatímco je produkt CICS spuštěný.

Poznámka: Použití CEDA může způsobit chybu, pokud se některý z položek ve skupině momentálně používá.

Konec informací o programovacím rozhraní, které jsou citlivé na produkt.

Programování aplikací se sdílenými frontami

Toto téma poskytuje informace o některých faktorech, které je třeba vzít v úvahu při návrhu nových aplikací pro použití sdílených front a při migraci existujících aplikací do prostředí sdílené fronty.

Serializace aplikací

Určité typy aplikací možná budou muset zajistit, aby zprávy byly načítány z fronty přesně ve stejném pořadí, v jakém byly do fronty přijaty.

Je-li například IBM MQ použit ke stínování aktualizací databáze na vzdáleném systému, zpráva popisující aktualizaci záznamu musí být zpracována za zprávou popisující vložení tohoto záznamu. V prostředí lokálního řazení do fronty je to často dosaženo aplikací, která získává zprávy otevírající frontu s volbou MQOO_INPUT_EXCLUSIVE, čímž se zabrání jakémukoli jinému získání aplikace ze zpracování fronty ve stejnou dobu.

Produkt IBM MQ umožňuje aplikacím otevírat sdílené fronty výhradně stejným způsobem. Pokud však aplikace pracuje z oblasti fronty (například všechny aktualizace databáze jsou ve stejné frontě, ale ty, které mají být použity pro tabulku A a pro tabulku B je korelační identifikátor B), a aplikace chtějí získat zprávy pro aktualizace tabulky A a tabulky B souběžně, jednoduchý mechanizmus otevření fronty exkluzivně není možný.

Má-li tento typ aplikace využívat výhod vysoké dostupnosti sdílených front, můžete rozhodnout, že jiná instance aplikace, která přistupuje ke stejným sdíleným frontám, je spuštěna v sekundárním správci front, by měla převzít, pokud dojde k selhání primárního správce front nebo správce front.

Selže-li primární správce front, dojde k dvěma věcem:

- Obnovení sdílené fronty typu peer zajišťuje, že všechny nedokončené aktualizace z primární aplikace budou dokončeny nebo vráceny.
- Sekundární aplikace převezme zpracování fronty.

Sekundární aplikace může být spuštěna dříve, než budou vyřešeny všechny nedokončené jednotky práce, které by mohly vést k tomu, že sekundární aplikace načte zprávy mimo pořadí. Chcete-li vyřešit tento typ problému, může se aplikace rozhodnout jako *serializovaná aplikace*.

Serializovaná aplikace používá volání MQCONNX k připojení ke správci front a při připojení této aplikace určuje značku připojení, která je k této aplikaci jedinečná. Libovolné jednotky práce provedené aplikací jsou označeny značkou připojení. Produkt IBM MQ zajišťuje, aby jednotky práce v rámci skupiny sdílení front se stejnou značkou připojení byly serializovány (podle voleb serializace ve volání MQCONNX).

To znamená, že pokud primární aplikace používá volání MQCONNX se značkou připojení produktu `Database shadow retrievera` sekundární převzatá aplikace se pokusí použít volání MQCONNX s identickou značkou připojení, sekundární aplikace se nemůže připojit ke druhému serveru IBM MQ , dokud nebudou dokončeny všechny zbývající primární jednotky práce, v tomto případě pomocí partnerského zotavení.

Zvažte použití aplikační techniky serializované aplikace pro aplikace, které závisejí na přesné posloupnosti zpráv ve frontě. Konkrétně se jedná o následující podmínky:

- Aplikace, které se nesmí restartovat po selhání aplikace nebo správce front, dokud nejsou dokončeny všechny operace potvrzení a vrácení pro předchozí provedení aplikace.

  V takovém případě je serializovaná technika aplikace použitelná pouze v případě, že aplikace pracuje v synchronizačním bodu.

- Aplikace, které se nesmí spustit, je-li již spuštěna jiná instance stejné aplikace.

  V takovém případě je serializovaná technika aplikace vyžadována pouze v případě, že aplikace nemůže otevřít frontu pro výhradní vstup.

Poznámka: IBM MQ zaručuje zachování posloupnosti zpráv pouze v případě, že jsou splněna určitá kritéria. Ty jsou popsány v popisu příkazu MQGET.

Aplikace, které nejsou vhodné pro použití se sdílenými frontami
Některé funkce produktu IBM MQ nejsou při používání sdílených front podporovány, takže aplikace, které tyto funkce používají, nejsou vhodné pro prostředí sdílené fronty.

Při návrhu aplikací ve sdílené frontě zvažte následující body:

- Indexování fronty je omezeno na sdílené fronty. Chcete-li použít identifikátor zprávy nebo identifikátor korelace k výběru zprávy, kterou chcete načíst z fronty, měla by být fronta indexována se správnou hodnotou. Pokud vybíráte zprávy pouze podle identifikátoru zprávy, fronta potřebuje typ indexu MQIT_MSG_ID (i když můžete také použít MQIT_NONE). Pokud vybíráte zprávy podle identifikátoru korelace samostatně, musí mít fronta typ index MQIT_CORREL_ID.
- Jako sdílené fronty nemůžete používat dočasné dynamické fronty. Můžete však použít trvalé dynamické fronty. Modely pro sdílené dynamické fronty mají hodnotu DEFTYPE SHAREDYN (sdílená dynamická), ačkoli jsou vytvořena a zničena stejným způsobem jako funkce PERMDYN (trvalé dynamické).

Rozhodování o tom, zda mají být sdíleny neaplikační fronty
Tyto informace použijte při zvažování sdílení neaplikačních front.

Existují jiné fronty než aplikační fronty, které byste mohli zvážit při sdílení:

Inicializační fronty

Pokud definujete sdílenou inicializační frontu, nemusíte mít monitor spouštěčů spuštěný ve všech správci front ve skupině sdílení front, pokud je spuštěn alespoň jeden monitor spouštěčů. (Můžete také použít sdílenou inicializační frontu, i když je monitor spouštěčů spuštěný na každém správci front ve skupině sdílení front.)

Máte-li sdílenou aplikační frontu a použijete typ spouštěče typu EVERY (nebo typ spouštěče FIRST s malým intervalem spouštěče, který se chová jako typ spouštěče EVERY), vaše inicializační fronta musí být vždy sdílená fronta. Další informace o tom, kdy použít sdílenou inicializační frontu, najdete v tématu Tabulka 116 na stránce 861.

SYSTEM.* fronty
Můžete definovat SYSTEM.ADMIN.* fronty použité k zadržení zpráv událostí jako sdílené fronty. To může být užitečné při kontrole vyrovnávání zátěže, pokud dojde k výjimce. Každá zpráva události vytvořená produktem IBM MQ obsahuje identifikátor korelace určující, který správce front jej vytvořil.

Musíte definovat SYSTEM.QSG fronty použité pro sdílené kanály a řazení do front v rámci skupiny jako sdílené fronty.

Můžete také změnit definice SYSTEM.DEFAULT.LOCAL.QUEUE , která má být sdílena, nebo definovat svou vlastní výchozí definici sdílené fronty. To je popsáno v sekci **Definování systémových objektů** v příručce Plánování v systému z/OS *IBM MQ for z/OS Concepts and Planning Guide*.

Nemůžete definovat žádnou jinou frontu SYSTEM.* je fronty jako sdílené fronty.

Migrace existujících aplikací pro použití sdílených front
Kódy příčiny, spuštění spouštěče a volání rozhraní API MQINQ mohou v prostředí sdílené fronty fungovat odlišně.

Migrace existujících front do sdílených front je popsána v příručce Administrace produktu IBM MQ for z/OS *IBM MQ for z/OS System Administration Guide*.

Při migraci existujících aplikací vezměte v úvahu následující skutečnosti, které by mohly fungovat jiným způsobem v prostředí sdílené fronty:

Kódy příčin
Při migraci existujících aplikací pro použití sdílených front zkontrolujte nové kódy příčiny, které lze vydat.

Spouątění
Pokud používáte sdílenou frontu aplikací, spouštění funguje pouze pro potvrzené zprávy (na nesdílené aplikační frontě, spouštění funguje na všech zprávách).

Pokud pro spouštění aplikací používáte spouštění, možná budete chtít použít sdílenou inicializační frontu. Část Tabulka 116 na stránce 861 popisuje, co je třeba zvážit při rozhodování o tom, jaký typ inicializační fronty použít.

Tabulka 116. Kdy použít frontu sdílených inicializací

	Nesdílená fronta aplikací	**Sdílená fronta aplikací**
Nesdílená inicializační fronta	Co se týče předchozích verzí.	Používáte-li typ spouštěče FIRST nebo DEPTH, můžete použít nesdílenou inicializační frontu se sdílenou aplikační frontou. Přebytečné zprávy spouštěče mohou být generovány, ale toto nastavení je dobré pro spouštění dlouhodobě spuštěných aplikací (jako CICS bridge) a poskytuje vysokou dostupnost. Pro typ spouštěče FIRST nebo DEPTH spouští spouštěcí zpráva instanci aplikace v každém správci front, který spouští monitor spouštěčů a který ještě nemá otevřenou frontu aplikace pro vstup. Pro každého správce front je vygenerována jedna zpráva spouštěče, pokud existuje více než jeden monitor spouštěčů spuštěných na nesdílené lokální inicializační frontě, v konkrétním správci front, budou soupeřit o zpracování zprávy.
Sdílená inicializační fronta	Nepoužívejte sdílenou inicializační frontu s nesdílenou frontou aplikací.	Máte-li sdílenou aplikační frontu, která má typ spouštěče EVERY, použijte sdílenou inicializační frontu nebo za určitých okolností můžete ztratit zprávy spouštěče, například selhání správce front. Pro typ spouštěče FIRST nebo DEPTH je jedna spouštěcí zpráva generována každým správcem front, který má otevřenou inicializační frontu otevřenou pro vstup. **Poznámka:** Pro typ spouštěče FIRST nebo DEPTH, pokud je jedna instance monitoru spouštěčů zaneprázdněna, ponechává tento potenciál pro méně vytížené monitory spouštěčů, aby zpracovaly více než jednu spouštěcí zprávu ze sdílené inicializační fronty. Proto může být spuštěno více instancí aplikace serveru proti danému správci front. Všimněte si, že tyto vícenásobné instance jsou spuštěny jako výsledek zpracování více zpráv spouštěče. Obvykle jde o typ spouštěče FIRST nebo DEPTH, pokud instance aplikace již obsluhuje frontu aplikací, další zprávu spouštěče nebude generovat správce front, k němuž je aplikace připojena.

MQINQ

Použijete-li volání `MQINQ` k zobrazení informací o sdílené frontě, hodnoty počtu volání `MQOPEN` , které mají otevřenou frontu pro vstup a výstup, se vztahují pouze ke správci front, který volání vydal. Ve skupině sdílení front, které mají otevřenou frontu, nejsou vytvářeny žádné informace o jiných správcích front.

z/OS Aplikace mostu IMS a IMS v systému IBM MQ for z/OS

Tyto informace vám pomohou při psaní aplikací produktu IMS pomocí produktu IBM MQ.

- Chcete-li použít synchronizační body a volání MQI v aplikacích produktu IMS , prohlédněte si téma “Zápis aplikací produktu IMS pomocí produktu IBM MQ” na stránce 58.
- Chcete-li psát aplikace, které používají most IBM MQ - IMS , přečtěte si téma “Zápis aplikací mostu IMS” na stránce 62.

Následující odkazy použijte k vyhledání dalších informací o aplikacích mostu IMS a IMS v systému IBM MQ for z/OS:

- “Zápis aplikací produktu IMS pomocí produktu IBM MQ” na stránce 58
- “Zápis aplikací mostu IMS” na stránce 62

Související pojmy

“Přehled rozhraní fronty zpráv” na stránce 690
Zde jsou uvedeny informace o komponentách rozhraní MQI (Message Queue Interface).

“Připojování k správci front a odpojování od něj” na stránce 704
Chcete-li používat programovací služby IBM MQ , musí mít program připojení ke správci front. V této části se dozvíte, jak se připojit k správci front a odpojit je od správce front.

“Otevírání a zavírání objektů” na stránce 712
Tyto informace poskytují vhled do otevírání a zavírání objektů IBM MQ .

“Vložení zpráv do fronty” na stránce 723
V této části se dozvíte, jak vložit zprávy do fronty.

“Získávání zpráv z fronty” na stránce 737
Tyto informace použijte k získání informací o získávání zpráv z fronty.

“Inquaning about a nastavení atributů objektu” na stránce 815
Atributy jsou vlastnosti, které definují charakteristiky objektu IBM MQ .

“Potvrzení a zálohování jednotek práce” na stránce 818
Tyto informace popisují, jak potvrdit a zazálohovat všechny zotavitelné operace get a put, které se vyskytly v jednotce práce.

“Spuštění aplikací produktu IBM MQ pomocí spouštěčů” na stránce 829
Informace o spouštěčích a o tom, jak spustit aplikace IBM MQ pomocí spouštěčů.

“Práce s MQI a klastry” na stránce 847
Existují speciální volby na voláních a návratových kódech, které se vztahují ke klastrování.

“Použití a zápis aplikací v systému IBM MQ for z/OS” na stránce 851
Aplikace produktu IBM MQ for z/OS mohou být vytvořeny z programů spuštěných v mnoha různých prostředích. To znamená, že mohou využít výhody zařízení dostupných ve více než jednom prostředí.

Zápis aplikací produktu IMS pomocí produktu IBM MQ

Při použití produktu IBM MQ v aplikacích produktu IMS jsou k dispozici další aspekty, které zahrnují volání rozhraní API produktu MQ a mechanizmus používaný pro synchronizační bod.

Následující odkazy použijte k vyhledání dalších informací o zápisu aplikací IMS v systému IBM MQ for z/OS:

- “Synchronizační body v aplikacích produktu IMS” na stránce 58
- “Volání MQI v aplikacích produktu IMS” na stránce 59

Omezení

Existují omezení, ve kterých volání rozhraní API produktu IBM MQ může používat aplikace s použitím adaptéru IMS .

Následující volání rozhraní API produktu IBM MQ nejsou v rámci aplikace s použitím adaptéru IMS podporována:

- MQCB
- FUNKCE MQCB_

- MQCTL

Související pojmy

“Zápis aplikací mostu IMS” na stránce 62
Toto téma obsahuje informace o zápisu aplikací pro použití mostu IBM MQ - IMS .

Synchronizační body v aplikacích produktu IMS
V aplikaci IMS zavedete synchronizační bod pomocí volání příkazu IMS , jako je GU (get unique) na IOPCB a CHKP (checkpoint).

Chcete-li odvolat všechny změny od předchozího kontrolního bodu, můžete použít volání IMS ROLB (rollback). Další informace najdete v následující dokumentaci:

- IMS 13 Application Programming APG SC19-3646
- IMS 13 Rozhraní API pro programování aplikací APR SC19-3647

Správce front je účastníkem protokolu s dvoufázovým potvrzováním; správce synchronizačního bodu IMS je koordinátorem.

Všechny otevřené manipulátory jsou uzavřeny adaptérem IMS v synchronizačním bodu (s výjimkou dávkového nebo neřízeného prostředí BMP). Důvodem je skutečnost, že jiný uživatel by mohl zahájit další jednotku práce a kontrola zabezpečení produktu IBM MQ se provádí při volání MQCONN, MQCONNX a MQOPEN, nikoli při volání MQPUT nebo MQGET.

V prostředí WFI (Wait-for-Input) nebo PWFI (pseudo Wait-for-Input) IMS však IBM MQ neuzavřete obslužné rutiny, dokud nebude doručena další zpráva nebo dokud nebude do aplikace vrácen stavový kód QC. Pokud aplikace čeká v oblasti IMS a některá z těchto obslužných rutin patří ke spouštěcím frontám, spuštění se neproběhne, protože fronty jsou otevřené. Z tohoto důvodu by aplikace spuštěné v prostředí WFI nebo PWFI měly explicitně MQCLOSE před provedením příkazu GU na port IOPCB pro další zprávu.

Je-li při volání MQDISC použita aplikace IMS (buď BMP nebo MPP), otevřené fronty jsou zavřeny, ale není proveden žádný implicitní synchronizační bod. Pokud aplikace skončí normálně, všechny otevřené fronty se zavřou a dojde k implicitnímu odevzdání. Pokud se aplikace ukončí nestandardně, všechny otevřené fronty se zavřou a dojde k implicitnímu odvolání.

Volání MQI v aplikacích produktu IMS
Tyto informace použijte k seznámení s použitím volání MQI v aplikacích serveru a v aplikacích Poptávka.

Tento oddíl pokrývá použití volání MQI v následujících typech aplikací produktu IMS :

- “Serverové aplikace” na stránce 863
- “Dotazové aplikace” na stránce 865

Serverové aplikace

Zde je obrys modelu aplikace serveru MQI:

```
Initialize/Connect
.
Open queue for input shared
.
Get message from IBM MQ queue
.
Do while Get does not fail
.
If expected message received
Process the message
Else
Process unexpected message
End if
.
Commit
.
Get next message from IBM MQ queue
.
End do
.
```

```
Close queue/Disconnect
.
END
```

Ukázkový program CSQ4ICB3 zobrazuje implementaci ve formátu BMP pomocí tohoto modelu v produktu C/370. Program nejprve naváže komunikaci s IMS a potom pomocí IBM MQ:

```
main()
----
Call InitIMS
If IMS initialization successful
Call InitMQM
If IBM MQ initialization successful
Call ProcessRequests
Call EndMQM
End-if
End-if

Return
```

Inicializace IMS určuje, zda byl program volán jako řízený zprávami nebo jako dávkový soubor typu BMP orientovaný na dávky a řídí příslušným způsobem připojení správce front IBM MQ a příslušné fronty:

```
InitIMS
-------
Get the IO, Alternate and Database PCBs
Set MessageOriented to true

Call ctdli to handle status codes rather than abend
If call is successful (status code is zero)
While status code is zero
Call ctdli to get next message from IMS message queue
If message received
Do nothing
Else if no IOPBC
Set MessageOriented to false
Initialize error message
Build 'Started as batch oriented BMP' message
Call ReportCallError to output the message
End-if
Else if response is not 'no message available'
Initialize error message
Build 'GU failed' message
Call ReportCallError to output the message
Set return code to error
End-if
End-if
End-while
Else
Initialize error message
Build 'INIT failed' message
Call ReportCallError to output the message
Set return code to error
End-if

Return to calling function
```

Inicializace IBM MQ se připojí ke správci front a otevře fronty. V souboru BMP s řízeným zprávami se volá po každém synchronním synchronizačním bodu IMS ; v souboru BMP s dávkovým zaměřením se toto volání volá pouze během spouštění programu:

```
InitMQM
-------
Connect to the queue manager
If connect is successful
Initialize variables for the open call
Open the request queue
If open is not successful
Initialize error message
Build 'open failed' message
Call ReportCallError to output the message
Set return code to error
End-if
Else
```

```
Initialize error message
Build 'connect failed' message
Call ReportCallError to output the message
Set return code to error
End-if

Return to calling function
```

Implementace modelu serveru v MPP je ovlivněna skutečností, že MPP zpracovává jedinou jednotku práce na vyvolání. Důvodem je to, že když je proveden synchronizační bod (GU), jsou uzavřeny obslužné rutiny připojení a fronty a je doručena další zpráva IMS . Toto omezení může být částečně překonán jedním z následujících způsobů:

- **Zpracování mnoha zpráv v rámci jedné jednotky-práce**

  To zahrnuje:

  – Čtení zprávy
  – Zpracování požadovaných aktualizací
  – Zadání odpovědi

  ve smyčce, dokud nebudou zpracovány všechny zprávy nebo dokud nezpracuje nastavený maximální počet zpráv, v tomto okamžiku se provede synchronizační bod.

  Touto cestou lze přistupovat pouze k určitým typům aplikací (například k aktualizaci jednoduché databáze nebo dotazu). Ačkoli lze zprávy s odpovědí rozhraní MQI odesílat s oprávněním původce zprávy modulu MQI, která je zpracovávána, je třeba pečlivě řešit důsledky zabezpečení pro všechny aktualizace prostředků produktu IMS .
- **Zpracovává se jedna zpráva na vyvolání MPP a zajišťuje více plánování MPP pro zpracování všech dostupných zpráv.**

  Pomocí programu monitoru spouštěčů produktu IBM MQ IMS (CSQQTRMN) můžete naplánovat transakci MPP, pokud ve frontě IBM MQ jsou zprávy a žádné aplikace, které ji obsluhují.

  Pokud monitor spouštěčů spustí MPP, jméno správce front a jméno fronty jsou předány programu, jak je zobrazeno v následujícím extraktu kódu COBOL:

```
* Data definition extract
01 WS-INPUT-MSG.
05 IN-LL1         PIC S9(3) COMP.
05 IN-ZZ1         PIC S9(3) COMP.
05 WS-STRINGPARM      PIC X(1000).
01 TRIGGER-MESSAGE.
COPY CMQTMC2L.
*
* Code extract
GU-IOPCB SECTION.
MOVE SPACES TO WS-STRINGPARM.
CALL 'CBLTDLI' USING GU,
IOPCB,
WS-INPUT-MSG.
IF IOPCB-STATUS = SPACES
MOVE WS-STRINGPARM TO MQTMC.
*  ELSE handle error
*
* Now use the queue manager and queue names passed
DISPLAY 'MQTMC-QMGRNAME     ='
MQTMC-QMGRNAME OF MQTMC '='.
DISPLAY 'MQTMC-QNAME       ='
MQTMC-QNAME  OF MQTMC '='.
```

Model serveru, který se očekává, že bude dlouhodobě spuštěnou úlohou, je lépe podporován v oblasti dávkového zpracování, ačkoli BMP nelze spustit pomocí CSQQTRMN.

Dotazové aplikace

Typická aplikace IBM MQ zahajující dotaz nebo aktualizace pracuje následujícím způsobem:

- Shromáždit data od uživatele

- Vložit jednu nebo více zpráv produktu IBM MQ
- Získejte zprávy odpovědi (možná budete muset na ně počkat)
- Zadejte odpověď uživateli.

Vzhledem k tomu, že zprávy vkládané do front produktu IBM MQ nejsou zpřístupněny ostatním aplikacím produktu IBM MQ , dokud nejsou potvrzeny, musí být buď uvedeny v synchronizačním bodu, nebo musí být aplikace IMS rozdělena do dvou transakcí.

Pokud dotaz zahrnuje vložení jedné zprávy, můžete použít volbu *bez synchronizačního bodu* ; pokud je však dotaz složitější nebo pokud jsou zahrnuty aktualizace prostředků, může dojít k problémům s konzistencí, dojde-li k selhání a nepoužijete syncpointing.

Chcete-li to překonat, můžete pomocí volání MQI rozdělit transakce MPP IMS pomocí volání MQI, viz téma *IMS/ESA Application Programming: Data Communication* , kde získáte informace o tomto tématu. To umožňuje implementovat dotazovací program v MPP:

```
Initialize first program/Connect
.
Open queue for output
.
Put inquiry to IBM MQ queue
.
Switch to second IBM MQ program, passing necessary data in save
pack area (this commits the put)
.
END
.
.
Initialize second program/Connect
.
Open queue for input shared
.
Get results of inquiry from IBM MQ queue
.
Return results to originator
.
END
```

Zápis aplikací mostu IMS

Toto téma obsahuje informace o zápisu aplikací pro použití mostu IBM MQ - IMS .

Informace o mostu IBM MQ - IMS naleznete v tématu Most systému IMS.

Následující odkazy použijte k vyhledání více informací o zápisu aplikací mostu IMS v systému IBM MQ for z/OS:

- “Jak se most IMS zabývá zprávami” na stránce 62
- “Zápis transakčních programů IMS prostřednictvím IBM MQ” na stránce 873

Související pojmy

“Zápis aplikací produktu IMS pomocí produktu IBM MQ” na stránce 58
Při použití produktu IBM MQ v aplikacích produktu IMS jsou k dispozici další aspekty, které zahrnují volání rozhraní API produktu MQ a mechanizmus používaný pro synchronizační bod.

Jak se most IMS zabývá zprávami

Pokud použijete most IBM MQ - IMS k odesílání zpráv do aplikace produktu IMS , je třeba vytvořit zprávy ve speciálním formátu.

Musíte také vložit zprávy do front produktu IBM MQ , které byly definovány s třídou úložiště, která určuje skupinu XCF a název člena cílového systému IMS . Jsou známy jako fronty mostu MQ-IMS nebo pouze fronty **mostu** .

Most IBM MQ-IMS vyžaduje výlučný vstupní přístup (MQOO_INPUT_EXCLUSIVE) do fronty mostu, je-li definován s QSGDISP (QMGR) nebo pokud je definován s QSGDISP (SHARED) spolu s volbou NOSHARE.

Uživatel se nemusí přihlašovat k produktu IMS před odesláním zpráv do aplikace IMS . ID uživatele v poli *UserIdentifier* struktury MQMD se používá pro kontrolu zabezpečení. Úroveň kontroly se určuje, když

se produkt IBM MQ připojí k produktu IMSa je popsán v tématu Řízení přístupu k aplikacím pro most IMS. To umožní implementaci pseudo přihlášení.

Most IBM MQ - IMS přijímá následující typy zpráv:

- Zprávy obsahující transakční data produktu IMS a strukturu MQIIH (popsané v části MQIIH):

```
MQIIH LLZZ<trancode><data>[LLZZ<data>][LLZZ<data>]
```

Poznámka:

1. Hranaté závorky, [], představují volitelné vícesegmentové segmenty.
2. Chcete-li použít strukturu MQIIH, nastavte pole *Format* struktury MQMD na MQFMT_IMS.

- Zprávy obsahující transakční data produktu IMS , ale ne strukturu MQIIH:

```
LLZZ<trancode><data> \
[LLZZ<data>][LLZZ<data>]
```

IBM MQ ověřuje data zprávy, aby se zajistilo, že součet bajtů LL plus délka MQIIH (je-li přítomna) se rovná délce zprávy.

Pokud most produktu IBM MQ - IMS získává zprávy z front mostu, zpracuje je následujícím způsobem:

- Obsahuje-li zpráva strukturu MQIIH, most ověřuje záhlaví MQIIH (viz MQIIH), sestaví záhlaví OTMA a odešle zprávu do produktu IMS. Kód transakce je uveden ve vstupní zprávě. Pokud se jedná o LTERM, IMS odpoví zprávou DFS1288E . Pokud kód transakce představuje příkaz, příkaz IMS provede tento příkaz; v opačném případě je zpráva zařazena do fronty v produktu IMS pro danou transakci.
- Pokud zpráva obsahuje transakční data produktu IMS , ale ne strukturu MQIIH, most IMS provede následující předpoklady:
  - Kód transakce je v bajtech 5 až 12 uživatelských dat.
  - Transakce se nachází v nekonverzačním režimu.
  - Transakce je v režimu vázaného zpracování 0 (operace commit-then-send)
  - *Format* v MQMD se používá jako *MFSMapName* (na vstupu).
  - Režim zabezpečení je MQISS_CHECK.

  Zpráva odpovědi je také sestavena bez struktury MQIIH a přebírá *Format* pro deskriptor MQMD z výstupu *MFSMapName* výstupu IMS .

Most IBM MQ - IMS používá jeden nebo dva Tpipes pro každou frontu IBM MQ :

- Synchronizovaná Tpipe se používá pro všechny zprávy pomocí režimu vázaného zpracování 0 (COMMIT_THEN_SEND) (tyto zobrazují se SYN ve stavovém poli u příkazu TPIPE xxxx produktu IMS /DIS TMEMBER)
- Nesynchronizovaná Tpipe se používá pro všechny zprávy pomocí režimu vázaného zpracování 1 (SEND_THEN_COMMIT)

Tpipe jsou vytvářeny produktem IBM MQ při jejich prvním použití. Nesynchronizovaná Tpipe existuje, dokud se produkt IMS nerestartuje. Synchronizované nástroje Tpipe existují, dokud produkt IMS nestuduje. Tyto Trubury nelze odstranit sami.

Další informace o tom, jak produkt IBM MQ - IMS pracuje se zprávami, naleznete v následujících tématech:

- “Mapování zpráv produktu IBM MQ na typy transakcí produktu IMS” na stránce 64
- “Pokud zprávu nelze vložit do fronty IMS” na stránce 64
- “Kódy zpětné vazby mostu IMS” na stránce 65
- “Pole MQMD ve zprávách z mostu IMS” na stránce 65
- “Pole MQIIH ve zprávách z mostu IMS” na stránce 66

- “Odpověď na zprávy z IMS” na stránce 67
- “Použití PCB s alternativním obsahem v transakcích produktu IMS” na stránce 67
- “Odesílání nevyžádaných zpráv z produktu IMS” na stránce 67
- “Segmentace zpráv” na stránce 68
- “Převod dat” na stránce 68

Související pojmy

“Zápis transakčních programů IMS prostřednictvím IBM MQ” na stránce 873

Kódování vyžadované pro zpracování transakcí produktu IMS prostřednictvím produktu IBM MQ závisí na formátu zprávy, který je vyžadován transakcí IMS a o rozsahu odpovědí, které může vrátit. Existuje však několik bodů, které byste měli zvážit, když vaše aplikace zpracovává informace o formátování obrazovky produktu IMS .

Mapování zpráv produktu IBM MQ na typy transakcí produktu IMS

Tabulka popisující mapování zpráv produktu IBM MQ na typy transakcí produktu IMS .

Tabulka 117. Mapování zpráv produktu IBM MQ na typy transakcí produktu IMS

IBM MQ typ zprávy	**Potvrzení-then-send (režim 0)-používá synchronizované IMS Tpipe**	**Odeslat-pak-potvrdit (režim 1)-používá nesynchronizované IMS Tpipe**
Trvalé zprávy IBM MQ	• Obnovitelné transakce plné funkce • Nezotavitelné transakce jsou odmítnuty IMS	• Transakce Fastpath • Konverzační transakce • Transakce s plnou funkcí
Netrvalé zprávy IBM MQ	• Nezotavitelné transakce plné funkce • Napravitelné transakce jsou povoleny s IMS V8 a APAR PQ61404 a všemi pozdějšími verzemi IMS	• Transakce Fastpath • Konverzační transakce • Transakce s plnou funkcí

Poznámka: Příkazy IMS nemohou používat trvalé zprávy IBM MQ s režimem vázaného zpracování 0. Další informace naleznete v příručce *IMS/ESA Open Transaction Manager Access User's Guide* .

Pokud zprávu nelze vložit do fronty IMS

Informace o akcích, které mají být provedeny v případě, že zprávu nelze vložit do fronty produktu IMS .

Pokud zprávu nelze vložit do fronty IMS , provede příkaz IBM MQnásledující akce:

- Pokud nelze zprávu vložit do IMS , protože zpráva je neplatná, zpráva se umístí do fronty nedoručených zpráv a zpráva se odešle na systémovou konzolu.
- Je-li zpráva platná, ale je odmítnuta produktem IMS, IBM MQ pošle na systémovou konzolu chybovou zprávu, zpráva obsahuje chybový kód IMS a zpráva IBM MQ se umístí do fronty nedoručených zpráv. Pokud má chybový kód IMS hodnotu 001A, IMS pošle zprávu IBM MQ obsahující příčinu selhání odpovědi na frontu pro odpověď.

  Poznámka: Za výše uvedených okolností, pokud IBM MQ nemůže z nějakého důvodu vložit zprávu do fronty nedoručených zpráv, je zpráva vrácena do původní fronty produktu IBM MQ . Do systémové konzoly se odešle chybová zpráva a z této fronty se neodesílají žádné další zprávy.

  Chcete-li zprávy znovu odeslat, proveďte ***jednu*** z následujících možností:

  – Zastavte a restartujte Tpipe v IMS odpovídající frontě
  – Změňte frontu na GET (DISABLED), a znovu na GET (ENABLED)
  – Zastavte a znovu spusťte IMS nebo OTMA

  - Zastavte a znovu spusťte svůj subsystém IBM MQ .
- Je-li zpráva odmítnuta IMS pro cokoli jiného, než je chyba zprávy, vrátí se zpráva IBM MQ do původní fronty, IBM MQ zastaví zpracování fronty a chybová zpráva se odešle na systémovou konzolu.

  Je-li vyžadována zpráva o výjimce, most ji vloží do fronty pro odpověď s oprávněním původce. Pokud zpráva nemůže být vložena do fronty, zpráva sestavy se umístí do fronty nedoručených zpráv s oprávněním mostu. Pokud ji nelze umístit do fronty DLQ, je vyřazena.

Kódy zpětné vazby mostu IMS
Kódy příčiny IMS jsou obvykle výstupem v hexadecimálním formátu ve zprávách konzoly produktu IBM MQ , jako je například CSQ2001I (například, chybový kód 0x001F). IBM MQ kódů zpětné vazby, jak je vidět v záhlaví zpráv vložených do fronty nedoručených zpráv, jsou desítková čísla.

Kódy zpětné vazby mostu IMS jsou v rozsahu 301 až 399, nebo 600 až 855 pro chybový kód NACK 0x001A. Jsou namapovány z chybových kódů IMS-OTMA takto:

1. Zjištěný kód IMS-OTMA se převede z hexadecimálního čísla na desítkové číslo.
2. Do čísla, které je výsledkem výpočtu v 1, je přidáno 300, což dává kód IBM MQ *Feedback* .
3. Smysl IMS-OTMA 0x001A, desetinné číslo 26 je speciální případ. Je vygenerován kód *Feedback* v rozsahu 600-855.

   a. Kód příčiny IMS-OTMA se převede z hexadecimálního čísla na desítkové číslo.

   b. 600 je přidáno k číslu, které je výsledkem výpočtu v a, což poskytuje kód IBM MQ *Feedback* .

Informace o chybových kódech služby IMS-OTMA najdete v tématu Chybové kódy OTMA pro zprávy NAK.

Pole MQMD ve zprávách z mostu IMS
Informace o polích MQMD ve zprávách z mostu IMS .

MQMD z původní zprávy je přenášeno IMS v sekci Uživatelská data záhlaví OTMA. Pokud zpráva pochází z produktu IMS, je tato zpráva vytvořena uživatelskou procedurou rozpoznání místa určení IMS . MQMD zprávy přijaté od produktu IMS je sestaveno takto:

StrucID
"MD"

Verze
MQMD_VERSION_1

Sestava
MQRO_NONE

MsgType
MQMT_REPLY

Vypršení
Je-li hodnota MQIIH_PASS_EXPIRATION nastavena v poli Příznaky MQIIH, obsahuje toto pole zbývající čas vypršení platnosti, jinak je nastaven na hodnotu MQEI_UNLIMITED.

Zpětná vazba
MQFB_NONE

Kódování
MQENC.Native (kódování systému z/OS)

CodedCharSetId
MQCCSI_Q_MGR (CodedCharSetID) systému z/OS)

Formát
MQFMT_IMS, je-li MQMD.Format vstupní zprávy je MQFMT_IMS, jinak IOPCB.MODNAME

Priorita
MQMD.Priority vstupní zprávy

Trvání
Závisí na režimu vázaného zpracování: MQMD.Persistence vstupní zprávy, pokud perzistence CM-1; odpovídá zotavitelnosti zprávy IMS , pokud CM-0

MsgId
MQMD.MsgId , pokud MQRO_PASS_MSG_ID, jinak nové MsgId (výchozí)

CorrelId
MQMD.CorrelId ze vstupní zprávy, je-li MQRO_PASS_CORREL_ID, jinak MQMD.MsgId ze vstupní zprávy (předvolba)

BackoutCount
0

ReplyToQ
Mezery

ReplyToQMgr
Mezery (nastavení na lokální název správce front qmgr správcem front během operace MQPUT)

UserIdentifier
MQMD.UserIdentifier vstupní zprávy

AccountingToken
MQMD.AccountingToken vstupní zprávy.

ApplIdentityData
MQMD.ApplIdentityData vstupní zprávy

PutApplType
MQAT_XCF, pokud není žádná chyba, jinak MQAT_BRIDGE

PutApplName
<XCFgroupName> <XCFmemberName> není-li žádná chyba, jinak název QMGR

PutDate
Datum, kdy byla zpráva vložena

PutTime
Čas, kdy byla zpráva vložena

ApplOriginData
Mezery

Pole MQIIH ve zprávách z mostu IMS
Získejte informace o polích MQIIH ve zprávách z mostu IMS .

MQIIH zprávy přijaté z produktu IMS je sestaveno takto:

StrucId
"IIH"

Verze
1

StrucLength
84

Kódování
MQENC_NATIVE

CodedCharSetId
MQCCSI_Q_MGR

Formát
MQIIH.ReplyToFormat vstupní zprávy, pokud MQIIH.ReplyToFormat není prázdný, jinak IOPCB.MODNAME

Příznaky
0

LTermOverride
Název LTERM (Tpipe) z hlavičky OTMA

MFSMapName
Název mapování z záhlaví OTMA

Formát ReplyTo
Mezery

Ověřovatel
MQIIH.Authenticator vstupní zprávy, je-li zpráva odpovědi vložena do fronty mostu MQ-IMS , jinak prázdné.

ID TranInstance
Konverzační ID/token serveru z hlavičky OTMA, pokud se jedná o konverzaci. Ve verzích systému IMS starších než V14je toto pole vždy prázdné, pokud se nejedná o konverzaci. V systému IMS V14 může být toto pole nastaveno systémem IMS i v případě, že se nejedná o konverzaci.

TranState
"C", je-li v konverzaci, jinak prázdné

CommitMode
Režim vázaného zpracování v záhlaví OTMA ("0" nebo "1")

SecurityScope
Prázdný

Vyhrazené
Prázdný

Odpověď na zprávy z IMS
Když je transakce systému IMS transakcí ISRTs na její IPCB, zpráva je směrována zpět na původní LTERM nebo TPIPE.

Tyto zprávy jsou v produktu IBM MQ zobrazeny jako zprávy odpovědí. Zprávy odpovědí z produktu IMS jsou vloženy do fronty pro odpovědi určené v původní zprávě. Pokud zpráva nemůže být vložena do fronty pro odpověď, je umístěna do fronty nedoručených zpráv s použitím autority mostu. Pokud zprávu nelze umístit do fronty nedoručených zpráv, odešle se do produktu IMS negativní potvrzení, které potvrdí, že zprávu nelze přijmout. Odpovědnost za zprávu se pak vrátí do IMS. Pokud používáte režim vázaného zpracování 0, zprávy z této Tpipe se neodesílají na most a zůstanou ve frontě IMS . To znamená, že se neodešlou žádné další zprávy, dokud nebude restartován. Používáte-li režim vázaného zpracování 1, může pokračovat jiná práce.

Pokud má odpověď strukturu MQIIH, je její typ formátu MQFMT_IMS; pokud ne, je tento typ formátu určen pomocí názvu IMS MOD použitého při vkládání zprávy.

Použití PCB s alternativním obsahem v transakcích produktu IMS
Když transakce IMS používá PCB s alternativním odezvou (ISRTs na ALTPCB nebo vydává volání CHNG na modifikovatelný PCB), vyvolá se uživatelská procedura před směrováním (DFSYPRX0), aby určila, zda by zpráva měla být přesměrována.

Má-li být zpráva přesměrována, je pro potvrzení cíle vyvolána uživatelská procedura rozpoznání místa určení (DFSYDRU0) a informace o hlavičce uvádí téma Použití uživatelských procedur OTMA v produktu IMS a DFSYPRX0 pro informace o těchto ukončovacích programech.

Pokud se akce neprovedou ve výstupních procedurách, veškerý výstup z transakcí produktu IMS zahájených ze správce front produktu IBM MQ bez ohledu na to, zda má být IOPCB nebo ALTPCB, bude vrácen do stejného správce front.

Odesílání nevyžádaných zpráv z produktu IMS
Chcete-li odesílat zprávy z produktu IMS do fronty produktu IBM MQ , je třeba vyvolat transakci IMS , která má hodnotu ISRTs na ALTPCB.

You need to write pre-routing and destination resolution exits to route unsolicited messages from IMS and build the OTMA user data, so that the MQMD of the message can be built correctly. Informace

o těchto ukončovacích programech najdete v tématu Ukončení předběžného směrování DFSYPRX0 a Uživatelská procedura rozpoznání místa určení .

Poznámka: Most IBM MQ - IMS neví, zda je zpráva, kterou obdrží, odpověď nebo nevyžádaná zpráva. To zpracovává stejnou zprávu v každém případě, sestavení MQMD a MQIIH odpovědi založené na OTMA UserData , která byla doručena se zprávou.

Nevyžádané zprávy mohou vytvářet nové Tpipe. Například pokud se existující transakce IMS přepnula na nový LTERM (například PRINT01), ale implementace vyžaduje, aby byl výstup dodán prostřednictvím OTMA, vytvoří se nové Tpipe (s názvem PRINT01 v tomto příkladu). Standardně se jedná o nesynchronizovanou Tpipe. Vyžaduje-li implementace zprávu, která má být obnovitelná, nastavte výstupní příznak ukončení rozlišení místa určení. Další informace viz příručka *IMS Customization Guide* .

Segmentace zpráv
Můžete definovat transakce IMS jako očekávaný vstup z jednoho nebo více segmentů.

Původní aplikace IBM MQ musí vytvořit uživatelský vstup za strukturu MQIIH jako jeden nebo více segmentů dat LLZZ. Všechny segmenty zprávy IMS musí být obsaženy v jediné zprávě IBM MQ odeslané s jedním MQPUT.

Maximální délka segmentu LLZZ-datového segmentu je definována systémem IMS/OTMA (32767 bajtů). Celková délka zprávy IBM MQ je součtem bajtů LL a délky struktury MQIIH.

Všechny segmenty odpovědi jsou obsaženy v jediné zprávě IBM MQ .

Pro zprávy ve formátu MQFMT_IMS_VAR_STRING existuje další omezení pro omezení velikosti 32 kB. Když se data ve zprávě ASCII-smíšená CCSID převedou na zprávu CCSID se smíšeným EBCDIC, je bajt shift-in nebo shift-out bajt přidán pokaždé, když dojde k přechodu mezi SBCS a znaky DBCS. Omezení 32 KB se použije na maximální velikost zprávy. To znamená, že pole LL ve zprávě nesmí přesáhnout 32 kB, zpráva nesmí překročit 32 kB včetně všech znaků shift-in a shift-out. Verze aplikace, kterou tato zpráva musí mít, musí umožňovat toto.

Převod dat
Převod dat provádí buď prostředek distribuovaného systému front (který může volat všechny nezbytné uživatelské procedury), nebo agent fronty v rámci skupiny (který nepodporuje použití uživatelských procedur) při vložení zprávy do cílové fronty, která má informace o XCF definované pro příslušnou paměťovou třídu. Konverze dat se nevyskytne, když je zpráva doručena do fronty prostřednictvím publikování/odběru.

Všechny potřebné uživatelské procedury musí být k dispozici pro prostředek distribuovaných front v datové sadě, na kterou odkazuje příkaz CSQXLIB DD. To znamená, že můžete odesílat zprávy do aplikace IMS pomocí mostu IBM MQ - IMS z libovolné platformy IBM MQ .

Pokud došlo k chybám převodu, je zpráva vložena do fronty bez převodu; výsledkem je nakonec, že most produktu IBM MQ - IMS je považován za chybu, protože most nemůže rozpoznat formát záhlaví. Dojde-li k chybě konverze, odešle se na konzolu z/OS chybová zpráva.

Podrobné informace o převodu dat obecně najdete v příručce “Zápis uživatelských procedur pro převod dat” na stránce 948 .

Odesílání zpráv na most IBM MQ - IMS

Chcete-li zajistit, aby byl převod proveden správně, musíte správci front sdělit, jaký má formát zprávy.

Pokud má zpráva strukturu MQIIH, musí být *Format* v MQMD nastaveno na vestavěný formát MQFMT_IMS a produkt *Format* v MQIIH musí být nastaven na název formátu, který popisuje data zprávy. Pokud zde není MQIIH, nastavte *Format* v deskriptoru MQMD na název vašeho formátu.

Jsou-li vaše data (jiná než LLZs) všechna znaková data (MQCHAR), použijte jako název vašeho formátu (v záhlaví MQIIH nebo MQMD) vestavěný formát MQFMT_IMS_VAR_STRING. Jinak použijte jméno vašeho formátu, v takovém případě musíte také poskytnout uživatelskou proceduru pro převod dat pro svůj formát. Výjezd musí obsloužit konverzi LLZs ve vaší zprávě, kromě samotných dat (ale nemusí se zpracovávat žádné MQIIH na začátku zprávy).

Pokud vaše aplikace používá produkt *MFSMapName*, můžete místo toho použít zprávy s rozhraním MQFMT_IMS a definovat název mapování předaný transakci IMS v poli MFSMapName MQIIH.

Příjem zpráv z mostu IBM MQ - IMS

Je-li na původní zprávě, kterou odesíláte do produktu IMS, přítomna struktura MQIIH, ve zprávě s odpovědí se také nachází jedna z nich.

Chcete-li se ujistit, že je vaše odpověď správně převedena:

- Pokud máte ve své původní zprávě strukturu MQIIH, uveďte formát, který chcete pro svou zprávu odpovědi, v poli MQIIH *ReplytoFormat* původní zprávy. Tato hodnota je umístěna v poli MQIIH *Format* zprávy odpovědi. To je zvláště užitečné, pokud všechna vaše výstupní data jsou ve tvaru LLZZ < znaková data >.
- Pokud ve vaší původní zprávě nemáte strukturu MQIIH, uveďte formát, který chcete pro zprávu odpovědi jako název MFS MOD v ISRT aplikace IMS na IOVNOSE.

Zápis transakčních programů IMS prostřednictvím IBM MQ
Kódování vyžadované pro zpracování transakcí produktu IMS prostřednictvím produktu IBM MQ závisí na formátu zprávy, který je vyžadován transakcí IMS a o rozsahu odpovědí, které může vrátit. Existuje však několik bodů, které byste měli zvážit, když vaše aplikace zpracovává informace o formátování obrazovky produktu IMS .

Je-li transakce IMS spuštěna z obrazovky 3270, zpráva se předá prostřednictvím služby IMS Message Format Services. To může odebrat veškerou závislost na terminálu z datového toku, který je v transakci vidět. Je-li transakce spuštěna přes OTMA, MFS se neúčastní. Je-li aplikační logika implementována v systému MFS, musí být znovu vytvořena v nové aplikaci.

V některých transakcích produktu IMS může aplikace koncového uživatele upravit určité chování obrazovky 3270, například zvýraznění pole, které má zadaná neplatná data. Tento typ informací se sděluje přidáním dvoubajtového pole atributu do zprávy IMS pro každé pole obrazovky, které je třeba upravit programem.

Pokud tedy kóduji aplikaci k napodobení relace 3270, je třeba při sestavování nebo přijímání zpráv brát v úvahu tato pole.

Možná budete potřebovat kódovat informace ve vašem programu ke zpracování:

- Který klíč je stisknut (například Enter a PF1)
- Kde se kurzor nachází, když je zpráva předána do vaší aplikace
- Zda byla pole atributu nastavena aplikací IMS
  - Vysoká, normální nebo nulová intenzita
  - Barva
  - Určuje, zda produkt IMS očekává pole zpět při příštím stisknutí klávesy Enter.
- Určuje, zda má aplikace IMS v libovolném poli používat znaky null (X'3F').

Pokud vaše zpráva IMS obsahuje pouze znaková data (kromě datového segmentu LLZZ) a používáte strukturu MQIIH, nastavte formát MQMD na MQFMT_IMS a formát MQIIH na MQFMT_IMS_VAR_STRING.

Pokud vaše zpráva IMS obsahuje pouze znaková data (kromě datového segmentu LLZZ) a **nevyužíváš** strukturu MQIIH, nastavte formát MQMD na MQFMT_IMS_VAR_STRING a ujistěte se, že vaše aplikace IMS určuje při odpovídání volbu MODname MQFMT_IMS_VAR_STRING. Pokud se vyskytne problém (například uživatel není autorizován k použití transakce) a IMS odešle chybovou zprávu, má identifikátor MODname ve tvaru DFSMOx, kde x je číslo v rozsahu od 1 do 5. Tato funkce je vložena do deskriptoru MQMD.Format.

Pokud vaše zpráva IMS obsahuje binární data, balená data nebo data s plovoucí řádovou čárkou (kromě datového segmentu LLZZ), můžete kódovat vlastní rutiny pro převod dat. Informace o formátování obrazovky produktu IMS naleznete v příručce *IMS/ESA Application Programming: Transaction Manager* .

Při psaní kódu pro zpracování transakcí produktu IMS prostřednictvím produktu IBM MQzvažte následující témata.

Zápis aplikací IBM MQ k vyvolání konverzačních transakcí IMS

Tyto informace použijte jako vodítko pro uvážení při zápisu aplikace IBM MQ pro vyvolání konverzačních transakcí produktu IMS .

Když zapíšete aplikaci, která vyvolává konverzaci IMS , zvažte následující:

- Zahrnout strukturu MQIIH do zprávy aplikace.
- Nastavte *CommitMode* v MQIIH na MQICM_SEND_THEN_COMMIT.
- Chcete-li vyvolat novou konverzaci, nastavte *TranState* v MQIIH na MQITS_NOT_IN_CONVERSATION.
- Chcete-li vyvolat druhý a následující postup konverzace, nastavte *TranState* na hodnotu MQITS_IN_CONVERSATION a nastavte *TranInstanceId* na hodnotu pole vráceného v předchozím kroku konverzace.
- V produktu IMS neexistuje žádný snadný způsob, jak zjistit hodnotu *TranInstanceId*, pokud byste ztratili původní zprávu odeslanou z produktu IMS.
- Aplikace musí zkontrolovat *TranState* zpráv z IMS , aby zkontrolována, zda transakce IMS ukončila konverzaci.
- Chcete-li ukončit konverzaci, můžete použít /EXIT. Musíte také citovat *TranInstanceId*, nastavit *TranState* na hodnotu MQITS_IN_CONVERSATION a použít frontu IBM MQ , na které se konverzace provádí.
- Pro zadržení nebo uvolnění konverzace nelze použít /HOLD nebo /REL.
- Konverzace vyvolaná prostřednictvím mostu IBM MQ - IMS jsou ukončeny, pokud je produkt IMS restartován.

Psaní programů obsahujících příkazy IMS

Aplikační program může sestavit zprávu IBM MQ ve tvaru LLZZ*příkaz*namísto transakce, kde *příkaz* je ve formě /DIS TRAN PART nebo /DIS POOL ALL.

Většina příkazů IMS může být vydávána tímto způsobem; podrobnosti naleznete v publikaci *IMS V11 Communications and Connections* . Výstup příkazu se přijímá ve zprávě s odezvou IBM MQ v textovém tvaru, jak by bylo odesláno na terminál 3270 pro zobrazení.

OTMA implementoval speciální formu příkazu display transaction IMS , který vrací navrhovanou formu výstupu. Přesný formát je definován v *IMS V11 Komunikace a připojení*. Chcete-li vyvolat tento formulář ze zprávy IBM MQ , sestavte data zprávy jako dříve, například /DIS TRAN PART, a nastavte pole TranState v MQIIH na MQITS_ARCHITECTED. Příkaz IMS zpracovává příkaz a vrací odpověď ve formě architektury. Odezva s architekturou obsahuje všechny informace, které lze najít v textové podobě výstupu, a další informace o tom, zda je transakce definována jako obnovitelná nebo neobnovitelná.

Spouątění

Most IBM MQ - IMS nepodporuje zprávy spouštěče.

Pokud definujete inicializační frontu, která používá paměťovou třídu s parametry XCF, budou zprávy odeslané do této fronty odmítnuty, jakmile se dostanou k mostu.

Zápis procedurálních aplikací klienta

Co potřebujete vědět, chcete-li psát klientské aplikace na serveru IBM MQ pomocí procedurálního jazyka.

Aplikace mohou být sestaveny a spouštěny v prostředí klienta IBM MQ . Aplikace musí být sestavena a propojena s použitou IBM MQ MQI client . Způsob, jakým jsou aplikace sestaveny a vzájemně propojeny,

se liší podle použité platformy a programovacího jazyka. Informace o tom, jak vytvářet klientské aplikace, viz “Sestavování aplikací pro produkt IBM MQ MQI clients” na stránce 880.

Aplikaci IBM MQ lze spustit jak v úplném prostředí produktu IBM MQ , tak v prostředí produktu IBM MQ MQI client , aniž byste změnili svůj kód za předpokladu, že jsou splněny určité podmínky. Další informace o spouštění aplikací v prostředí klienta produktu IBM MQ naleznete v tématu “Spuštění aplikací v prostředí produktu IBM MQ MQI client” na stránce 883.

Pokud použijete rozhraní MQI (Message Queue Interface) k zápisu aplikací ke spuštění v prostředí produktu IBM MQ MQI client , existují některé další ovládací prvky k uložení během volání MQI, aby bylo zajištěno, že zpracování aplikace IBM MQ není narušeno. Další informace o těchto ovládacích prvcích naleznete v tématu “Použití modulu MQI v klientské aplikaci” na stránce 875.

Informace o přípravě a spuštění jiných typů aplikací jako klientských aplikací naleznete v následujících tématech:

- “Příprava a spuštění aplikací CICS a Tuxedo” na stránce 894
- “Příprava a spuštění aplikací serveru Microsoft Transaction Server” na stránce 43
- “Příprava a spuštění aplikací produktu IBM MQ JMS” na stránce 897

Související pojmy

“Koncepty vývoje aplikací” na stránce 6
K zápisu aplikací IBM MQ můžete použít volbu procedurálních nebo objektově orientovaných jazyků. Odkazy v tomto tématu použijte pro informace o koncepcích produktu IBM MQ , které jsou užitečné pro vývojáře aplikací.

“Vyvíjení aplikací pro IBM MQ” na stránce 5
Můžete vyvíjet aplikace k odesílání a přijímání zpráv a ke správě správců front a souvisejících prostředků. IBM MQ podporuje aplikace napsané v mnoha různých jazycích a rámcích.

“Aspekty návrhu pro aplikace produktu IBM MQ” na stránce 43
Když se rozhodnete, jak vaše aplikace mohou využívat výhody platforem a prostředí, které máte k dispozici, musíte se rozhodnout, jak používat funkce nabízené produktem IBM MQ.

“Psaní procedurální aplikace pro řazení do fronty” na stránce 689
Prostřednictvím těchto informací můžete získat informace o vytváření aplikací pro řazení do front, připojování a odpojování od správce front, publikování a odběru a otevírání a zavírání objektů.

“Zapisování aplikací typu publikování/odběr” na stránce 774
Začněte psát aplikace pro publikování/odběr aplikací produktu IBM MQ .

“Sestavení procedurální aplikace” na stránce 966
Aplikaci IBM MQ můžete psát v jednom z několika procedurálních jazyků a aplikaci spustit na několika různých platformách.

“Obsluha chyb procedurálních programů” na stránce 1015
Tyto informace vysvětlují chyby související s voláními MQI MQI buď při volání, nebo při doručení její zprávy do konečného cíle.

Související úlohy

“Použití ukázkových procedurálních programů produktu IBM MQ” na stránce 1034
Tyto ukázkové programy jsou napsány v procedurálních jazycích a demonstrují typická použití rozhraní MQI (Message Queue Interface). Programy produktu IBM MQ na různých platformách.

“Vývoj webových služeb pomocí produktu IBM MQ” na stránce 1255
Můžete vyvíjet aplikace produktu IBM MQ pro webové služby pomocí přenosu IBM MQ pro protokol SOAP.

Použití modulu MQI v klientské aplikaci

Tato kolekce témat se zabývá rozdíly mezi zápisem vaší aplikace IBM MQ pro spuštění v prostředí klienta rozhraní MQI (MQI) a ke spuštění v prostředí úplného správce front produktu IBM MQ .

Při návrhu aplikace zvažte, jaké ovládací prvky je třeba uložit během volání MQI, abyste se ujistili, že zpracování aplikace IBM MQ není narušeno.

Před spuštěním aplikací, které používají rozhraní MQI, je třeba vytvořit určité objekty produktu IBM MQ . Další informace naleznete v tématu Aplikační programy používající rozhraní MQI.

Omezení velikosti zprávy v aplikaci klienta

Správce front má maximální délku zprávy, ale maximální velikost zprávy, kterou můžete přenášet z klientské aplikace, je omezena definicí kanálu.

Atribut maximální délky zprávy (MaxMsgLength) správce front je maximální délka zprávy, kterou může tento správce front zpracovat.

Multi V systému Multiplatformymůžete zvýšit atribut maximální délky zprávy správce front. Další informace viz ALTER QMGR.

Hodnotu proměnné MaxMsgpro správce front můžete zjistit pomocí volání MQINQ.

Pokud se změní atribut Délka MaxMsg, nekontroluje se, zda již neexistují fronty a dokonce zprávy s délkou větší než nová hodnota. Poté, co změníte tento atribut, restartujte aplikace a kanály, abyste se ujistili, že změna nabyla platnosti. Není tedy možné, aby byly generovány nové zprávy, které překročí délku MaxMsgDélka správce front nebo fronty (pokud není povolena segmentace správce front).

Maximální délka zprávy v definici kanálu omezuje velikost zprávy, kterou můžete přenášet po připojení klienta. Pokud se aplikace IBM MQ pokusí použít volání MQPUT nebo MQGET se zprávou větší než je tato, vrátí se do aplikace chybový kód. Parametr maximální velikosti zprávy definice kanálu neovlivňuje maximální velikost zprávy, kterou lze spotřebovat pomocí funkce MQCB v rámci připojení klienta.

Související pojmy

“Použití MQCONNX” na stránce 880
Volání MQCONNX můžete použít k určení struktury definice kanálu (MQCD) ve struktuře MQCNO.

Související informace

Maximální délka zprávy (MAXMSGL)
ZMĚNIT KANÁL
2010 (07DA) (RC2010): MQRC_DATA_LENGTH_ERROR

Výběr CCSID klienta nebo serveru

Pro klienta použijte lokální identifikátor kódované znakové sady (CCSID). Správce front provádí nezbytný převod. Použijte proměnnou prostředí MQCCSID k přepsání CCSID. Pokud vaše aplikace provádí více PUTs, pole CCSID a kódování deskriptoru MQMD lze přepsat po dokončení prvního PUT.

Data předaná v rámci rozhraní fronty zpráv (MQI) z aplikace do stubu klienta musí být v lokálním CCSID kódovaném pro IBM MQ MQI client. Vyžaduje-li připojený správce front data, která mají být převedena, provede tento převod kód podpory klienta ve správci front.

V produktu IBM WebSphere MQ 7.0 a v novějších verzích může klient produktu Java provést převod, pokud to správce front není schopen provést. Viz “IBM MQ classes for Java připojení klientů” na stránce 332.

Kód klienta předpokládá, že znaková data, která přecházejí z rozhraní MQI v klientovi, jsou v CCSID konfigurovaném pro danou pracovní stanici. Pokud tento CCSID není podporovaným CCSID nebo není vyžadovaným identifikátorem CCSID, může být přepsán pomocí proměnné prostředí MQCCSID pomocí jednoho z těchto příkazů:

- Windows

```
SET MQCCSID=850
```

- UNIX

```
export MQCCSID=850
```

- IBM i

```
ADDENVVAR ENVVAR(MQCCSID) VALUE(37)
```

Je-li tento parametr nastaven v profilu, předpokládá se, že všechna data MQI jsou v kódové stránce 850.

Poznámka: Předpokladem o kódové stránce 850 se nevztahují na data aplikace ve zprávě.

Pokud vaše aplikace provádí více operací PUTs, které zahrnují záhlaví IBM MQ po deskriptoru zpráv (MQMD), uvědomte si, že pole CCSID a kódování deskriptoru MQMD se přepíší po dokončení prvního PUT.

Po prvním PUT obsahují tato pole hodnotu, kterou používá připojený správce front k převodu záhlaví IBM MQ . Ujistěte se, že aplikace resetuje hodnoty na hodnoty, které vyžaduje.

Použití MQINQ v klientovi aplication

Některé hodnoty dotazované pomocí MQINQ jsou upraveny kódem klienta.

CCSID
: je nastaven na hodnotu CCSID klienta, nikoli na straně správce front.

MaxMsgLength
: je zredukován, pokud je omezen definicí kanálu. To bude nižší z těchto hodnot:

- Hodnota definovaná v definici fronty, nebo
- Hodnota definovaná v definici kanálu

Další informace najdete v tématu MQINQ.

Použití koordinace bodu synchronizace v aplikaci klienta

Aplikace spuštěná na základním klientovi může vydávat MQCMIT a MQBACK, ale rozsah řízení synchronizačního bodu je omezen na prostředky MQI. Externí správce transakcí můžete použít s rozšířeným transakčním klientem.

V rámci produktu IBM MQje jednou z rolí správce front ovládací prvek bodu synchronizace v rámci aplikace. Je-li aplikace spuštěna na základním klientovi produktu IBM MQ , může zadat MQCMIT a MQBACK, ale rozsah řízení synchronizačního bodu je omezen na prostředky MQI. Příkaz IBM MQ verb MQBEGIN není platný v základním prostředí klienta.

Aplikace spuštěné v úplném prostředí správce front na serveru mohou koordinovat více prostředků (například databází) prostřednictvím monitoru transakcí. Na serveru můžete použít Monitor transakcí dodávaný s produkty IBM MQ nebo jiný monitor transakcí, jako např. CICS. Monitor transakcí nemůžete používat s aplikací základního klienta.

Externí správce transakcí můžete použít s rozšířeným transakčním klientem IBM MQ . Viz Co je rozšířený transakční klient? pro podrobnosti.

Použití dopředného čtení v aplikaci klienta

Čtení napřed můžete použít na klientovi, abyste umožnili odeslání netrvalých zpráv klientovi, aniž by aplikace klienta musela požadovat zprávy.

Když klient vyžaduje zprávu ze serveru, odešle požadavek na server. Odešle samostatný požadavek pro každou zprávu, kterou spotřebuje. Chcete-li zvýšit výkon klientů, kteří spotřebovávají přechodné zprávy, tím, že se neodešlou tyto zprávy požadavku, může být klient nakonfigurován tak, aby používal dopředné čtení. Čtení napřed umožňuje odesílání zpráv klientovi, aniž by aplikace musela požadovat jejich zpracování.

Použití dopředného čtení může zlepšit výkon při spotřebovávání přechodných zpráv z klientské aplikace. Toto zlepšení výkonu je dostupné pro aplikace MQI i JMS . Klientské aplikace používající příkaz MQGET nebo asynchronní spotřeba využívají zlepšení výkonu při spotřebovávání přechodných zpráv.

Při volání MQOPEN s parametrem MQOO_READ_AHEAD povolí klient IBM MQ čtení napřed pouze v případě, že jsou splněny určité podmínky. Tyto podmínky zahrnují:

- Jak klient, tak i vzdálený správce front musí být IBM WebSphere MQ 7 nebo novější.
- Aplikace klienta musí být kompilována a propojena s použitím podprocesových knihoven klienta IBM MQ MQI.
- Kanál klienta musí používat protokol TCP/IP.
- Kanál musí mít nenulové nastavení SharingConversations (SHARECNV) v definici kanálu klienta i serveru.

Je-li povoleno čtení napřed, jsou zprávy odesílány do vyrovnávací paměti na klientovi s názvem vyrovnávací paměť dopředného čtení. Klient má vyrovnávací paměť dopředného čtení pro každou frontu, kterou má otevřené s povolenou dopředným čtením. Zprávy ve vyrovnávací paměti dopředného čtení nejsou trvalé. Klient pravidelně aktualizuje server informacemi o množství dat, které spotřeboval.

Ne všechny návrhy aplikací klienta jsou vhodné pro použití čtení napřed, protože ne všechny volby jsou podporovány pro použití. Je-li povoleno čtení napřed, musí být některé volby konzistentní mezi voláními MQGET. Pokud klient změní svá kritéria výběru mezi voláními MQGET, zprávy ukládané do vyrovnávací paměti dopředného čtení zůstávají uvízlé v paměti dopředného čtení klienta dopředného čtení. Další informace naleznete v tématu “Zlepšení výkonu přechodných zpráv” na stránce 754

Konfigurace dopředného čtení je řízena třemi atributy, MaximumSize, PurgeTimea UpdatePercentage, které jsou uvedeny ve stanze MessageBuffer konfiguračního souboru klienta IBM MQ .

Použití asynchronního vložení v aplikaci klienta

Při použití asynchronního vložení může aplikace vložit zprávu do fronty, aniž by čekala na odezvu správce front. Tuto akci můžete použít ke zlepšení výkonu systému zpráv v některých situacích.

Za normálních okolností, když aplikace vloží zprávu nebo zprávy do fronty pomocí příkazu MQPUT nebo MQPUT1, musí aplikace čekat na to, aby správce front potvrdil, že zpracoval požadavek MQI. Můžete zvýšit výkon systému zpráv, a to zejména u aplikací, které používají vazby klienta, a aplikací, které nasadíte velký počet malých zpráv do fronty, tím, že vyberete místo toho, že chcete asynchronně vkládat zprávy. Když aplikace asynchronně odešle zprávu, správce front nevrátí úspěch nebo selhání každého volání, ale můžete namísto toho zkontrolovat chyby pravidelně.

Chcete-li asynchronně vložit zprávu do fronty, použijte volbu MQPMO_ASYNC_RESPONSE v poli *Options* struktury MQPMO.

Pokud zpráva není vhodná pro asynchronní vložení, je vložena do fronty synchronně.

Při požadavku na asynchronní odezvu vložení pro volání MQPUT nebo MQPUT1nemusí CompCode a příčina MQCC_OK a MQRC_NONE nezbytně znamenat, že zpráva byla úspěšně vložena do fronty. Ačkoli úspěšné nebo neúspěšné ukončení jednotlivých volání MQPUT nebo MQPUT1 nemusí být okamžitě vráceno, první chyba, která se vyskytla při asynchronním volání, může být později určena voláním MQSTAT.

Podrobnější informace o MQPMO_ASC_NC_RESPONSE najdete v tématu Volby MQPMO.

Ukázkový program Asynchronous Put demonstruje některé funkce, které jsou k dispozici. Podrobné informace o funkcích a návrhu programu a o jeho spuštění najdete v tématu “Ukázkový program Asynchronous Put” na stránce 1053.

Použití sdílení konverzací v aplikaci klienta

V prostředí, ve kterém je povoleno sdílení konverzací, mohou konverzace sdílet instanci kanálu MQI.

Sdílení konverzací je řízeno dvěma poli s názvem SharingConversations, z nichž jedna je součástí struktury definice kanálu (MQCD) a jedna z nich je součástí struktury výstupního parametru kanálu (MQCXP). Pole SharingConversations na disku MQCD je celočíselná hodnota určující maximální počet konverzací, které mohou sdílet instanci kanálu přidruženou k kanálu. Pole SharingConversations v MQCXP je logická hodnota označující, zda je instance kanálu momentálně sdílená.

V prostředí, ve kterém není povoleno sdílení konverzací, nová připojení klienta s určujícím identickým rozhraním MQCD nebudou sdílet instanci kanálu.

Nové připojení klientské aplikace bude sdílet instanci kanálu, jsou-li splněny následující podmínky:

- Konce připojení klienta i připojení k serveru jsou konfigurovány pro sdílení konverzací a tyto hodnoty nejsou potlačeny ukončením kanálu.
- Hodnota MQCD pro připojení klienta (zadaná v rámci volání MQCONNX klienta nebo z tabulky CCDT) přesně odpovídá hodnotě MQCD připojení klienta zadané v rámci volání MQCONNX klienta nebo v tabulce CCDT, když byla poprvé zavedena existující instance kanálu. Všimněte si, že původní objekt MQCD mohl být následně změněn ukončením nebo dohadování kanálu, ale že porovnání je provedeno proti hodnotě, která byla dodána na klientský systém před provedením těchto změn.
- Limit konverzací sdílení na straně serveru není překročen.

Pokud se nové připojení klientské aplikace shoduje s kritérii pro spuštění sdílení instance kanálu s jinými konverzacemi, je toto rozhodnutí provedeno dříve, než se k této konverzaci zavolají všechny uživatelské procedury. Uživatelské procedury v této konverzaci nemohou změnit skutečnost, že sdílí instanci kanálu s jinými konverzací. Nejsou-li k dispozici žádné existující instance kanálu odpovídající nové definici kanálu, bude připojena nová instance kanálu.

Vyjednávání kanálů probíhá pouze pro první konverzaci v instanci kanálu; vyjednané hodnoty pro instanci kanálu jsou v této fázi fixní a nelze je změnit při zahájení následných konverzací. Ověření TLS se také provede pouze pro první konverzaci.

Je-li hodnota MQCD SharingConversations během inicializace zabezpečení ukončena, odešle nebo přijme ukončení pro první konverzaci na soketu buď na připojení klienta, nebo na konci spojení mezi serverem a připojením serveru, bude inicializována nová hodnota po všech těchto ukončovacích procedurách, aby bylo možné určit hodnotu konverzací sdílení pro instanci kanálu (nejnižší hodnota bude mít přednost).

Je-li vyjednaná hodnota pro sdílení konverzací nula, instance kanálu se nikdy nesdílí. Další ukončovací programy, které nastavují toto pole na nulu, se také spouštějí na své vlastní instanci kanálu.

Je-li vyjednaná hodnota pro sdílení konverzací větší než nula, pak je MQCXP SharingConversations nastaveno na hodnotu TRUE pro následná volání ukončení, což znamená, že jiné uživatelské programy na této instanci kanálu lze zadat souběžně s tímto voláním.

Při zápisu ukončovacího programu kanálu zvažte, zda bude spuštěn na instanci kanálu, která může zahrnovat sdílení konverzací. Pokud by instance kanálu mohla zahrnovat sdílení konverzací, uvažujte o vlivu na jiné instance kanálu uživatelské procedury pro změnu polí MQCD. Všechna pole MQCD mají společné hodnoty ve všech konverzacích sdílení. Pokud se po vytvoření instance kanálu pokusí změnit pole MQCD, mohou se setkat s problémy, protože jiné instance ukončovacích programů spuštěných na instanci kanálu by se mohly pokoušet o změnu stejných polí ve stejnou dobu. Pokud by tato situace mohla nastat s vašimi ukončovacími programy, musíte serializovat přístup ke kódu MQCD ve svém kódu ukončení.

Pokud pracujete s kanálem, který je definován pro sdílení konverzací, ale nechcete, aby se sdílení stalo na určité instanci kanálu, nastavte hodnotu MQCD SharingConversations na 1 nebo 0, když inicializujete kanál na první konverzaci na instanci kanálu. Viz SharingConversations , kde najdete vysvětlení hodnot SharingConversations.

Příklad

Sdílení konverzací je povoleno.

Používáte definici kanálu připojení klienta, která uvádí výstupní program.

Při prvním spuštění tohoto kanálu pozmění uživatelský program některé z parametrů MQCD, když je inicializován. Tyto akce jsou zpracovávány kanálem, takže definice, se kterou kanál běží, se nyní liší od té, která byla původně dodána. Parametr SharingConversations MQCXP je nastaven na hodnotu TRUE.

Při příštím připojení aplikace k tomuto kanálu je konverzace spuštěna na instanci kanálu, která byla spuštěna dříve, protože má stejnou původní definici kanálu. Instance kanálu, ke které se aplikace připojí podruhé, je stejná instance jako při prvním připojení k aplikaci. V důsledku toho používá definice, které byly změněny ukončovacím programem. Když je výstupní program inicializován pro druhou konverzaci, přestože může pozměnit pole MQCD, nepostupuje se podle kanálu. Tyto charakteristiky se vztahují na všechny následující konverzace, které sdílejí instanci kanálu.

Použití MQCONNX

Volání MQCONNX můžete použít k určení struktury definice kanálu (MQCD) ve struktuře MQCNO.

To umožňuje volající aplikaci klienta určit definici kanálu připojení klienta za běhu. Další informace naleznete v tématu Použití struktury MQCNO při volání MQCONNX. Když použijete MQCONNX, volání vydané na serveru závisí na úrovni serveru a konfiguraci modulu listener.

Když používáte MQCONNX z klienta, jsou ignorovány následující volby:

- VAZBA MQCNO_STANDARD_BINDING
- VAZBA MQCNO_FASTPATH_BINDING

Struktura MQCD, kterou lze použít, závisí na počtu verzí produktu MQCD, které používáte. Informace o verzích produktu MQCD (MQCD_VERSION) naleznete v tématu Verze MQCD. Strukturu MQCD můžete použít například k předávání programů uživatelských procedur na server. Používáte-li produkt MQCD verze 3 nebo novější, můžete použít strukturu k předání pole uživatelských procedur na server. Tuto funkci můžete použít k provedení více než jedné operace na stejné zprávě, jako je šifrování a komprese, přidáním ukončení pro každou operaci, spíše než úpravou existující uživatelské procedury. Pokud ve struktuře MQCD nezadáte žádné pole, budou zkontrolována jednotlivá výstupní pole. Další informace o programech výstupních bodů kanálu naleznete v části “Kanály-uživatelské programy pro kanály systému zpráv” na stránce 926.

Sdílené popisovače připojení na MQCONNX

Mezi různými podprocesy ve stejném procesu můžete sdílet manipulátory s použitím sdílených manipulátorů připojení.

Při zadání sdílené obslužné rutiny připojení může být manipulátor připojení vrácený z volání MQCONNX předán v následných voláních MQI na libovolném podprocesu v procesu.

Poznámka: Pro připojení ke správci front serveru, který nepodporuje sdílené obslužné rutiny připojení, můžete použít manipulátor sdíleného připojení na serveru IBM MQ MQI client .

Další informace viz “Použití MQCONNX” na stránce 880.

Sestavování aplikací pro produkt IBM MQ MQI clients

Aplikace mohou být sestaveny a spouštěny v prostředí produktu IBM MQ MQI client . Aplikace musí být sestavena a propojena s použitou IBM MQ MQI client . Způsob, jakým jsou aplikace sestaveny a vzájemně propojeny, se liší podle použité platformy a programovacího jazyka.

Má-li být aplikace spuštěna v prostředí klienta, můžete ji zapsat do jazyků zobrazených v následující tabulce:

Tabulka 118. Programovací jazyky podporované v prostředí klienta

Platforma klienta	C	JAZYK C++	COBOL	pTAL	RPG	Visual Basic
AIX	Ano	Ano	Ano			
HP-UX	Ano	Ano	Ano			
IBM i	Ano		Ano		Ano	
Linux	Ano	Ano	Ano			
Solaris	Ano	Ano	Ano			
Windows	Ano	Ano	Ano			Ano

Multi *Propojení aplikací jazyka C s kódem produktu IBM MQ MQI client*

Poté, co jste napsali aplikaci IBM MQ , kterou chcete spustit na serveru IBM MQ MQI client, musíte ji propojit s kódem IBM MQ MQI client .

Aplikaci můžete propojit se svým kódem IBM MQ MQI client dvěma způsoby:

1. Přímo připojením vaší aplikace ke správci front, v takovém případě musí být správce front na stejném počítači jako vaše aplikace.
2. Do souboru knihovny klienta, který vám poskytuje přístup ke správcům front na stejném počítači nebo na jiném počítači.

Produkt IBM MQ poskytuje soubor knihovny klienta pro každé prostředí:

AIX **AIX**

Knihovnu libmqic.a pro aplikace bez podprocesů nebo knihovnu libmqic_r.a pro aplikace s podprocesy.

HP-UX **HP-UX**

Knihovna libmqic.sl pro aplikace bez podprocesů, nebo knihovna libmqic_r.sl pro aplikace s podprocesy.

Linux **Linux**

Knihovnu libmqic.so pro aplikace bez podprocesů, nebo knihovnu libmqic_r.so pro aplikace s podprocesy.

IBM i **IBM i**

Svázat aplikaci klienta s klientským servisním programem LIBMQIC pro aplikace bez podprocesů, nebo servisní program LIBMQIC_R pro aplikace s podprocesy.

Solaris **Solaris**

libmqic.so.

Chcete-li používat programy na počítači, který má nainstalované pouze IBM MQ MQI client for Solaris , musíte překompilovat programy, abyste je propojovali s knihovnou klienta:

```
$ /opt/SUNWspro/bin/cc -o prog_name prog_name c -mt -lmqic \
-lsocket -lc -lnsl -ldl
```

Parametry musí být zadány ve správném pořadí, jak je zobrazeno.

Windows **Windows**

MQIC32.LIB.

ULW *Propojení aplikací C++ s kódem IBM MQ MQI client*

Můžete psát aplikace, které se mají spustit na klientovi v C + +. Metody sestavení se liší v závislosti na prostředí.

Informace o tom, jak propojit vaše aplikace C + +, najdete v tématu Vytváření programů v jazyce C++ pro IBM MQ.

Podrobné informace o všech aspektech použití jazyků C + + naleznete v tématu Použití jazyka C++

Multi *Propojení aplikací jazyka COBOL s kódem produktu IBM MQ MQI client*

Po napsání aplikace v jazyce COBOL, kterou chcete spustit na serveru IBM MQ MQI client, je třeba ji propojit s příslušnou knihovnou.

Produkt IBM MQ poskytuje soubor knihovny klienta pro každé prostředí:

AIX **AIX**
Propojte aplikaci COBOL bez podprocesů s knihovnou libmqicb.a nebo s aplikací s podporou podprocesů v jazyce COBOL s parametrem libmqicb_r.a.

HP-UX **HP-UX**
Propojte aplikaci COBOL bez podprocesů s knihovnou libmqicb.sl nebo s aplikací v jazyce COBOL s podporou podprocesů libmqicb_r.sl.

HP-UX **Linux**
Propojte aplikaci COBOL bez podprocesů s knihovnou libmqicb.so nebo s aplikací v jazyce COBOL s podporou podprocesů libmqicb_r.so.

IBM i **IBM i**
Svázat aplikaci klienta jazyka COBOL se servisním programem AMQCSTUB pro aplikace bez podprocesů nebo servisní program AMQCSTUB_R pro aplikace s podprocesy.

Solaris **Solaris**
Propojte aplikaci COBOL bez podprocesů s knihovnou libmqicb.so nebo s aplikací v jazyce COBOL s podporou podprocesů libmqicb_r.so.

Windows **Windows**
Propojte kód aplikace s knihovnou MQICCBB pro 32bitový jazyk COBOL. Produkt IBM MQ MQI client for Windows nepodporuje 16bitový COBOL.

Windows *Propojení aplikací produktu Visual Basic s kódem produktu IBM MQ MQI client*

You can link Microsoft Visual Basic applications with the IBM MQ MQI client code on Windows.

V 9.0.0 V produktu IBM MQ 9.0je podpora produktu Microsoft Visual Basic 6.0 zamítnuta. Třídy IBM MQ pro .NET jsou doporučenými náhradními technologiemi. Další informace viz Vývoj aplikací .NET.

Propojte aplikaci Visual Basic s následujícími soubory začlenění:

CMQB.bas
MQI

CMQBB.bas
MQAI

CMQCFB.bas
příkazy PCF

CMQXB.bas
Kanály

Nastavte `mqtype=2` pro klienta v kompilátoru Visual Basic , abyste zajistili správný automatický výběr knihovny DLL klienta:

MQIC32.dll
Windows 7, Windows 8, Windows 2008 a Windows 2012

Související pojmy

“Kódování v produktu Visual Basic” na stránce 1029
Informace, které je třeba zvážit při kódování programů IBM MQ v produktu Microsoft Visual Basic. Produkt Visual Basic je podporován pouze v systému Windows.

“Příprava programů produktu Visual Basic v produktu Windows” na stránce 998
Informace, které je třeba zvážit při používání programů produktu Microsoft Visual Basic v systému Windows.

Spuštění aplikací v prostředí produktu IBM MQ MQI client

Aplikaci IBM MQ lze spustit jak v úplném prostředí produktu IBM MQ , tak v prostředí produktu IBM MQ MQI client , aniž byste změnili svůj kód za předpokladu, že jsou splněny určité podmínky.

Tyto podmínky jsou následující:

- Aplikace se nemusí připojovat k více než jednomu správci front souběžně.
- Název správce front není uvozeno hvězdičkou (*) ve volání `MQCONN` nebo `MQCONNX` .
- Aplikace nemusí používat žádnou z výjimek uvedených v tématu Jaké aplikace běží na serveru IBM MQ MQI client?

Poznámka: Knihovny, které používáte při linkování, určují prostředí, ve kterém musí být aplikace spuštěna.

Při práci v prostředí produktu IBM MQ MQI client pamatujte na následující skutečnosti:

- Každá aplikace spuštěná v prostředí produktu IBM MQ MQI client má svá vlastní připojení k serverům. Aplikace vytvoří jedno připojení k serveru pokaždé, když vydá volání `MQCONN` nebo `MQCONNX` .
- Aplikace odesílá a přijímá zprávy synchronně. To znamená čekat mezi okamžikem, kdy je volání vydáno na klientovi, a návratem kódu dokončení a kódu příčiny v rámci sítě.
- Všechny konverze dat provádí server, ale viz také MQCCSID , kde jsou informace o přepisu nastavení CCSID počítače.

Připojování aplikací produktu IBM MQ MQI client ke správcům front

Aplikace běžící v prostředí produktu IBM MQ MQI client se může připojit ke správci front různými způsoby. Můžete použít proměnné prostředí, strukturu MQCNO, nebo tabulku definic klienta.

Když aplikace spuštěná v prostředí klienta produktu IBM MQ vydá volání MQCONN nebo MQCONNX, klient identifikuje, jak má být připojení navázáno. Je-li volání MQCONNX vydáno aplikací na klientovi IBM MQ , knihovna klienta MQI vyhledá informace o kanálu klienta v následujícím pořadí:

1. Pomocí obsahu polí *`ClientConnOffset`* nebo *`ClientConnPtr`* struktury MQCNO (je-li k dispozici). Tato pole identifikují strukturu definice kanálu (MQCD), která má být použita jako definice kanálu připojení klienta. Podrobnosti o připojení lze přepsat pomocí uživatelské procedury před připojením. Další informace viz “Odkazování na definice připojení pomocí předání před připojením z úložiště” na stránce 959.
2. Je-li nastavena proměnná prostředí MQSERVER, použije se kanál, který definuje.
3. Je-li definován soubor `mqclient.ini` a obsahuje parametry ServerConnectionParms, použije se kanál, který definuje. Další informace naleznete v tématu Konfigurace klienta pomocí konfiguračního souboru a stanza CHANNELS konfiguračního souboru klienta.
4. Jsou-li nastaveny proměnné prostředí MQCHLLIB a MQCHLTAB, použije se tabulka definic kanálů klienta, na kterou se bude odkazovat. V 9.0.0 Eventuálně z produktu IBM MQ 9.0poskytuje proměnná prostředí MQCCDTURL ekvivalentní funkci pro nastavení kombinace proměnných prostředí MQCHLLIB a MQCHLTAB. Je-li nastavena hodnota MQCCDTURL, použije se tabulka definic kanálů klienta, na kterou se odkazuje. Další informace naleznete v tématu Webový adresovatelný přístup k tabulce definic kanálů klienta.
5. Je-li definován soubor `mqclient.ini` a obsahuje atributy ChannelDefinitionDirectory a ChannelDefinitionFile, tyto atributy se používají k vyhledání tabulky definic kanálů klienta. Další informace naleznete v tématu Konfigurace klienta pomocí konfiguračního souboru a stanza CHANNELS konfiguračního souboru klienta.
6. Nakonec, nejsou-li proměnné prostředí nastaveny, klient vyhledá tabulku definic kanálů klienta s cestou a názvem, které jsou zavedeny ze souboru `DefaultPrefix` v souboru `mqs.ini` . Pokud vyhledávání v tabulce definic kanálů klienta selže, klient použije následující cesty:
   - Linux UNIX V systému UNIX and Linux: `/var/mqm/AMQCLCHL.TAB`
   - Windows V systému Windows: `C:\Program Files\IBM\MQ\amqclchl.tab`

- IBM i V systému IBM i: `/QIBM/UserData/mqm/@ipcc`
- V systému IBM MQ Appliance: *QMname*`_AMQCLCHL.TAB`. Objevují se pod `mqbackup:// URI`.

První z voleb popsaných v předchozím seznamu (s použitím polí *ClientConnOffset* nebo *ClientConnPtr* MQCNO) je podporováno pouze voláním MQCONNX. Pokud aplikace používá MQCONN místo MQCONNX, budou informace o kanálu vyhledány ve zbývajících pěti bodech v pořadí zobrazeném v seznamu. Pokud se klientovi nepodaří najít informace o kanálu, volání MQCONN nebo MQCONNX selže.

Název kanálu (pro připojení klienta) se musí shodovat s názvem kanálu připojení serveru definovaným na serveru pro volání MQCONN nebo MQCONNX, aby bylo možné provést úspěch.

Související informace

Tabulka definic kanálů klienta

V 9.0.0 Webový adresovatelný přístup k tabulce definic kanálů klienta

SERVER MQSERVER

MQCHLLIB

KARTA MQCHLTAB

V 9.0.0 MQCKTURTL.

Konfigurace připojení mezi serverem a klientem

Připojení klientských aplikací ke správcům front pomocí proměnných prostředí
Informace o kanálu klienta mohou být dodány aplikaci běžící v prostředí klienta pomocí proměnných prostředí.

Aplikace běžící v prostředí produktu IBM MQ MQI client se může připojit ke správci front s použitím následujících proměnných prostředí:

SERVER MQSERVER
Proměnná prostředí MQSERVER se používá k definování minimálního kanálu. Parametr MQSERVER určuje umístění serveru IBM MQ a komunikační metodu, která má být použita.

MQCHLLIB
Proměnná prostředí MQCHLLIB uvádí cestu k adresáři se souborem obsahujícím tabulku definic kanálů klienta (CCDT). Soubor je vytvořen na serveru, ale lze jej zkopírovat na pracovní stanici IBM MQ MQI client .

KARTA MQCHLTAB
Proměnná prostředí MQCHLTAB určuje název souboru, který obsahuje tabulku definic kanálů klienta (CCDT).

V 9.0.0 V produktu IBM MQ 9.0poskytuje proměnná prostředí MQCCDTURL ekvivalentní funkci pro nastavení kombinace proměnných prostředí MQCHLLIB a MQCHLTAB. Funkce MQCCDTURL umožňuje poskytovat soubor, ftp nebo adresu URL http jako jedinou hodnotu, ze které lze získat tabulku definic kanálů klienta. Další informace naleznete v tématu Webový adresovatelný přístup k tabulce definic kanálů klienta.

Připojení klientských aplikací ke správcům front pomocí struktury MQCNO
Definici kanálu můžete zadat ve struktuře definice kanálu (MQCD), která je dodána pomocí struktury MQCNO volání MQCONNX.

Další informace viz téma Použití struktury MQCNO při volání MQCONNX.

Připojení klientských aplikací ke správcům front pomocí tabulky definic kanálů klienta
Pokud použijete příkaz MQSC DEFINE CHANNEL, zadané podrobnosti se umístí do tabulky definic kanálů klienta (ccdt). Obsah parametru **QMgrName** volání MQCONN nebo MQCONNX určuje, ke kterému správci front se klient připojuje.

K tomuto souboru přistupuje klient, aby určil kanál, který bude aplikace používat. Existuje-li více než jedna vhodná definice kanálu, je volba kanálu ovlivněna atributy kanálu kanálu klienta (CLNTWGHT) a kanálu afinity připojení (AFFINITY).

Použití automatického opětovného připojení klienta
Můžete provést automatické opětovné připojení klientských aplikací, aniž byste zapisoval nějaký dodatečný kód, a to konfigurací počtu komponent.

Automatické opětovné připojení klienta je *vložené*. Toto připojení se automaticky obnoví v každém okamžiku aplikačního programu klienta a obnoví se všechny popisovače k otevřeným objektům.

Naopak ruční opětovné připojení vyžaduje, aby aplikace klienta znovu vytvořila připojení pomocí `MQCONN` nebo `MQCONNX` a znovu otevřela objekty. Automatické opětovné připojení klienta je vhodné pro řadu aplikací klienta, nikoliv však pro všechny.

Další informace najdete v tématu Automatické opětovné připojení klienta.

Role tabulky definic kanálů klienta
Tabulka definic kanálů klienta (CCDT) obsahuje definice kanálů připojení klienta. To je zvláště užitečné, pokud se vaše klientské aplikace mohou připojovat k řadě alternativních správců front.

Tabulka definic kanálů klienta se vytvoří, když definujete správce front. Stejný soubor může být použit více než jedním klientem IBM MQ .

Existuje řada způsobů, jak aplikaci klienta použít tabulku CCDT. Nástroje CCDT lze kopírovat na klientský počítač. CCDT můžete zkopírovat do umístění sdíleného více než jedním klientem. Nástroje CCDT lze zpřístupnit klientovi jako sdílený soubor, zatímco zůstává umístěn na serveru.

V 9.0.0 V produktu IBM MQ 9.0lze tabulku CCDT hostovat v centrálním umístění, které je přístupné prostřednictvím identifikátoru URI, a odebrat tak potřebu jednotlivě aktualizovat tabulku CCDT pro každého implementovaného klienta.

Související informace
Tabulka definic kanálů klienta

V 9.0.0 Webový adresovatelný přístup k tabulce definic kanálů klienta

Přístup k definicím kanálu připojení klienta

Skupiny správců front v tabulce CCDT
V tabulce definic kanálů klienta (CCDT) můžete definovat sadu připojení jako *skupinu správců front*. Aplikaci můžete připojit ke správci front, který je součástí skupiny správců front. To lze provést tak, že zafixujete název správce front na volání `MQCONN` nebo `MQCONNX` s hvězdičkou.

Můžete se rozhodnout definovat připojení k více než jednomu serverovém počítači, protože:

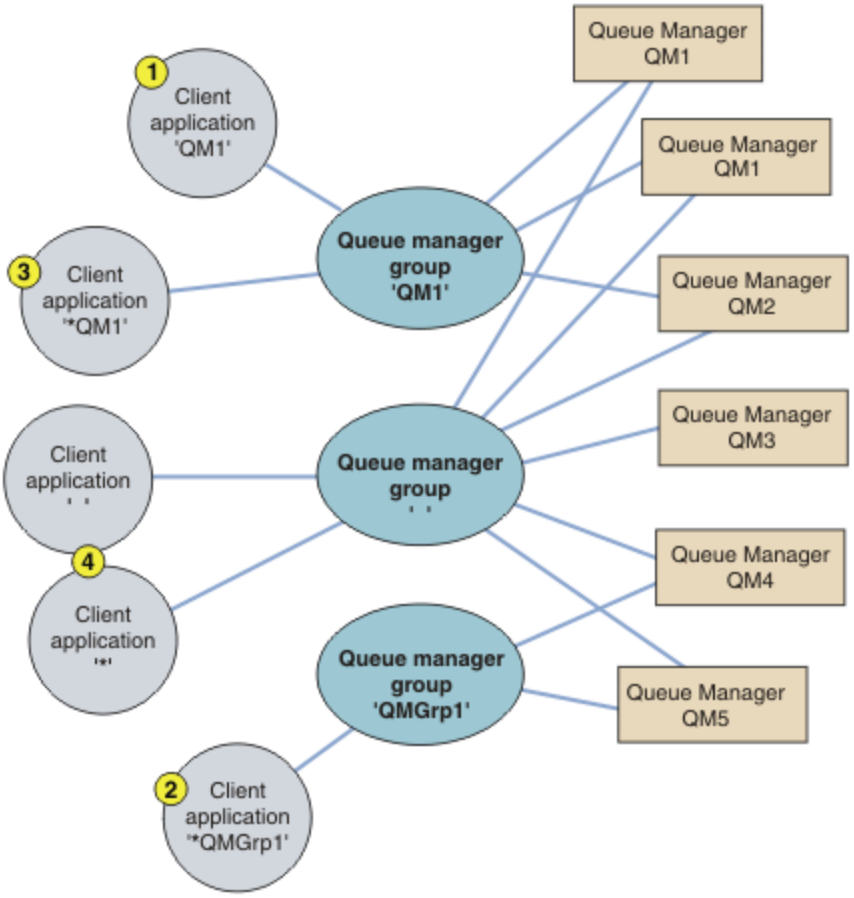
- Chcete-li zlepšit dostupnost, chcete připojit klienta k libovolné sadě správců front, kteří jsou spuštění.
- Chcete znovu připojit klienta ke stejnému správci front, k němuž došlo k úspěšnému připojení, ale připojit se k jinému správci front, pokud dojde k selhání připojení.
- Chcete-li znovu zkusit připojení klienta k jinému správci front, pokud dojde k selhání připojení, opětovným zadáním `MQCONN` v klientském programu chcete být znovu schopni připojení klienta k jinému správci front.
- Chcete automaticky znovu připojit připojení klienta k jinému správci front, pokud dojde k selhání připojení, aniž byste zapisoval nějaký kód klienta.
- Chcete-li automaticky znovu připojit připojení klienta k jiné instanci správce front s více instancemi, pokud dojde k selhání instance v pohotovostním režimu, bez zápisu jakéhokoliv kódu klienta.
- Chcete vyvážit připojení klienta přes několik správců front, s více klienty, kteří se připojují k některým správcům front, než u ostatních.
- Chcete rozložit opětovné připojení mnoha klientských připojení přes více správců front a v čase v případě, že vysoký objem připojení způsobí selhání.
- Chcete být schopni přesunout své správce front beze změny kódu aplikace klienta.
- Chcete zapsat aplikační programy klienta, které nepotřebují znát názvy správců front.

Není vždy vhodné připojit se k různým správcům front. Rozšířený transakční klient nebo klient Java v produktu WebSphere Application Servermůže například vyžadovat připojení k předvídatelné instanci správce front. Prostředí IBM MQ classes for Java nepodporuje automatické opětovné připojování klientů.

Skupina správců front je sada připojení definovaná v tabulce CCDT (Client Channel Definition table). Sada je definována svými členy se stejnou hodnotou atributu **QMNAME** ve svých definicích kanálů.

Obrázek 107 na stránce 886 je grafické znázornění tabulky připojení klienta zobrazující tři skupiny správců front, dvě pojmenované skupiny správců front zapsané v CCDT jako **QMNAME** (QM1) a **QMNAME** (QMGrp1) a jednu prázdnou nebo výchozí skupinu napsanou jako **QMNAME** ('').

1. Skupina správce front QM1 má tři kanály připojení klienta, které se připojují ke správcům front QM1 a QM2. QM1 může být správce front s více instancemi umístěný na dvou různých serverech.
2. Výchozí skupina správců front má šest kanálů připojení klienta připojujících se ke všem správcům front.
3. QMGrp1 má kanály připojení klienta ke dvěma správcům front, QM4 a QM5.



Obrázek 107. Skupiny správců front

Čtyři příklady použití této tabulky připojení klienta jsou popsány s pomocí očíslovaných klientských aplikací v produktu Obrázek 107 na stránce 886.

1. V prvním příkladu předá aplikace klienta název správce front QM1jako parametr **QmgrName** do jeho volání MQCONN nebo MQCONNX MQI. Kód klienta IBM MQ vybere odpovídající skupinu správců front QM1. Skupina obsahuje tři kanály připojení a produkt IBM MQ MQI client se pokusí připojit k serveru QM1 za použití každého z těchto kanálů, dokud nenajde modul listener produktu IBM MQ pro připojení připojené ke spuštěnému správci front s názvem QM1.

   Pořadí pokusů o připojení závisí na hodnotě atributu AFFINITY připojení klienta a na váze kanálu klienta. V rámci těchto omezení je pořadí pokusů o připojení randomizováno, a to jak přes tři možné připojení, tak v průběhu času, aby se rozšířily načítání vytvářeného připojení.

Volání MQCONN nebo MQCONNX vydané klientskou aplikací se zdaří, když je vytvořeno připojení k běžící instanci QM1.

2. Ve druhém příkladu aplikace klienta předává název správce front předponou s hvězdičkou, *QMGrp1 jako parametr **QmgrName** do jeho volání MQCONN nebo MQCONNX MQI. Klient IBM MQ vybere odpovídající skupinu správců front QMGrp1. Tato skupina obsahuje dva kanály připojení klienta a produkt IBM MQ MQI client se pokouší o připojení ke správci front *any* , který používá každý kanál. V tomto příkladu musí produkt IBM MQ MQI client vytvořit úspěšné připojení; název správce front, ke kterému se připojuje, je nepodstatné.

   Pravidlo pro pořadí pokusů o připojení je stejné jako u předchozích pokusů. Jediný rozdíl spočívá v tom, že při předběžném opravení názvu správce front s hvězdičkou klient indikuje, že název správce front není relevantní.

   Volání MQCONN nebo MQCONNX vydané aplikací klienta je úspěšné, když je vytvořeno připojení ke spuštěné instanci jakéhokoli správce front připojeného k kanálům ve skupině správců front QMGrp1 .

3. Třetí příklad je v podstatě stejný jako druhý, protože parametr **QmgrName** je vložen hvězdičkou, *QM1. Následující příklad ilustruje, že nelze určit, ke kterému správci front se má připojení kanálu klienta připojovat, a to tak, že zkontrolujete atribut QMNAME v rámci jedné definice kanálu. Skutečnost, že atribut **QMNAME** definice kanálu je QM1, nepostačuje k tomu, aby bylo možné vytvořit připojení ke správci front s názvem QM1. Pokud vaše klientská aplikace přečíslí svůj parametr **QmgrName** s hvězdičkou, pak je kterýkoli správce front možným cílem připojení.

   V tomto případě volání MQCONN nebo MQCONNX vydaná aplikací klienta uspějí, když je ustanoveno spojení se spuštěnou instancí buď QM1 , nebo QM2.

4. Čtvrtý příklad ilustruje použití výchozí skupiny. V takovém případě aplikace klienta předá hvězdičku, '*'nebo prázdnou hodnotu '', jako argument **QmgrName** do jeho volání MQCONN nebo MQCONNX MQI. Podle konvence v definici kanálu klienta označuje prázdný atribut **QMNAME** výchozí skupinu správců front a parametr **QmgrName** prázdný nebo hvězdička se shoduje s prázdným atributem **QMNAME** .

   V tomto příkladu má výchozí skupina správců front připojení kanálů klienta ke všem správcům front. Vyberete-li výchozí skupinu správců front, může být aplikace připojena k libovolnému správci front ve skupině.

   Volání MQCONN nebo MQCONNX vydané aplikací klienta se zdaří, když je vytvořeno připojení ke spuštěné instanci libovolného správce front.

   Poznámka: Výchozí skupina se liší od výchozího správce front, ačkoli aplikace používá prázdný parametr **QmgrName** pro připojení buď k výchozí skupině správců front, nebo k výchozímu správci front. Koncept výchozí skupiny správců front je relevantní pouze pro aplikaci klienta a pro výchozí správce front v aplikaci serveru.

Definujte kanály připojení klienta pouze u jednoho správce front, včetně těch kanálů, které se připojují ke druhému nebo třetímu správci front. Nedefinujte je ve dvou správcích front a poté se pokuste sloučit dvě definiční tabulky kanálu klienta. Klient může získat přístup pouze k jedné definiční tabulce kanálu klienta.

Příklady

Další informace naleznete v seznamu důvodů pro použití skupin správců front na začátku tématu. Jak pomocí skupiny správců front tyto možnosti poskytují?

Připojte se k některé z nich sady správců front.
Definujte skupinu správců front s připojeními ke všem správcům front v dané sadě a připojte se ke skupině pomocí parametru **QmgrName** s předponou hvězdičkou.

Znovu se připojte ke stejnému správci front, ale připojte se k jinému správci front, pokud je správce front připojen k poslední době nedostupný.
Definujte skupinu správců front jako dříve, ale nastavte atribut, **AFFINITY** (PREFERRED) na každé definici kanálu klienta.

Pokud připojení selže, zopakujte pokus o připojení k jinému správci front.
Připojte se ke skupině správců front a znovu zadejte volání MQCONN nebo MQCONNX MQI, je-li připojení přerušeno, nebo dojde k selhání správce front.

Automaticky se znovu připojit k jinému správci front, pokud připojení selže.
Připojte se ke skupině správců front pomocí volby MQCONNX **MQCNO** MQCNO_RECONNECT.

Automaticky se znovu připojte k jiné instanci správce front s více instancemi.
Postupujte stejně jako předchozí příklad. V tomto případě, chcete-li omezit skupinu správců front na připojení k instancím konkrétního správce front s více instancemi, definujte skupinu s připojeními pouze k instancím správce front s více instancemi.

Můžete také požádat klientskou aplikaci o vydání jeho volání MQCONN nebo MQCONNX MQI bez hvězdičky jako předpony s předponou **QmgrName** . Tímto způsobem se klientská aplikace může připojit pouze k uvedenému správci front. Nakonec můžete nastavit volbu **MQCNO** na hodnotu MQCNO_RECONNECT_Q_MGR. Tato volba přijímá nová připojení ke stejnému správci front, který byl dříve připojen. Tuto hodnotu můžete také použít k omezení opětovného připojení ke stejné instanci běžného správce front.

Vyrovnávání klientských připojení mezi správci front s dalšími klienty připojenými k některým správcům front než ostatním.
Definujte skupinu správců front a nastavte atribut **CLNTWGHT** na každé definici kanálu klienta, abyste nerovnoměrně distribuovali připojení.

Rozložit zátěž opakovaného připojení klienta nerovnoměrně a rozložit ji v čase po selhání připojení nebo správce front.
Postupujte stejně jako předchozí příklad. Funkce IBM MQ MQI client náhodně upravuje přepojení mezi správci front a přehojí opakované připojení v průběhu času.

Přesuňte správce front beze změny kódu klienta.
CCDT izoluje aplikaci klienta od umístění správce front. CCDT je datový soubor, který lze definovat na klientovi, číst ze sdíleného umístění nebo načíst z webového serveru. Další informace naleznete v tématu Tabulka definic kanálů klienta.

Napište aplikaci typu klient, která nezná názvy správců front.
Použijte názvy skupin správců front a stanovte konvence pojmenování pro názvy skupin správců front, které jsou relevantní pro vaše klientské aplikace ve vaší organizaci, a odráží spíše architekturu vašich řešení než pojmenování správců front.

z/OS *Připojování ke skupinám sdílení front*

Tuto aplikaci můžete připojit ke správci front, který je součástí skupiny sdílení front. To lze provést použitím názvu skupiny sdílení front namísto názvu správce front v rámci volání MQCONN nebo MQCONNX.

Názvy skupin sdílení front jsou tvořeny nejvýše čtyřmi znaky. Název musí být v síti jedinečný a nesmí být shodný s žádným názvem správce front.

Definice kanálu klienta by měla používat generické rozhraní skupiny sdílení front pro připojení k dostupnému správci front ve skupině. Další informace naleznete v tématu Připojení klienta ke skupině sdílení front. Je provedena kontrola, aby se zajistilo, že se správce front, ke kterému se modul listener připojuje, bude členem skupiny sdílení front.

Další informace o sdílených frontách naleznete v tématu Sdílené fronty a skupiny sdílení front.

Příklady váhy kanálu a afinity

Tyto příklady ilustrují, jak jsou kanály připojení klienta vybrány, když se použije nenulová hodnota ClientChannelWeights .

Atributy kanálu ClientChannel a ConnectionAffinity řídí způsob, jakým jsou kanály připojení klienta vybrány, je-li k dispozici více než jeden vhodný kanál pro připojení. Tyto kanály jsou konfigurovány pro připojení k různým správcům front za účelem zajištění vyšší dostupnosti, vyrovnávání pracovní zátěže nebo obojího. Volání MQCONN, která by mohla vést k připojení k jednomu z několika správců front, musí před názvem správce front uvést hvězdičku podle popisu v: Příklady volání MQCONN: Příklad 1. Název správce front obsahuje hvězdičku (*).

Použitelné kandidátské kanály pro připojení jsou ta, kde atribut QMNAME odpovídá názvu správce front uvedenému v rámci volání MQCONN. Pokud mají všechny vhodné kanály pro připojení hodnotu

ClientChannelVáha s hodnotou nula (výchozí), jsou tyto kanály vybrány v abecedním pořadí podle příkladu: Příklady volání MQCONN: Příklad 1. Název správce front obsahuje hvězdičku (*).

Následující příklady ilustrují, co se stane, když se použije nenulová hodnota ClientChannelWeights . Všimněte si, že vzhledem k tomu, že tato funkce zahrnuje pseudo-náhodný výběr kanálu, ukazují příklady posloupnost akcí, které se mohou stát, spíše než to, co rozhodně bude.

Příklad 1. Výběr kanálů, je-li parametr ConnectionAffinity nastaven na hodnotu PREFERRED
Tento příklad ilustruje, jak produkt IBM MQ MQI client vybere kanál z tabulky CCDT, kde je vlastnost ConnectionAffinity nastavena na hodnotu PREFERRED.

V tomto příkladu používá řada klientských počítačů tabulku CCDT (Client Channel Definition Table), kterou poskytuje správce front. Nástroje CCDT zahrnují kanály připojení klienta s následujícími atributy (zobrazené pomocí syntaxe příkazu DEFINE CHANNEL):

```
CHANNEL(A) QMNAME(DEV) CONNAME(devqm.it.company.example)
CHANNEL(B) QMNAME(CORE) CONNAME(core1.ops.company.example) CLNTWGHT(5) +
AFFINITY(PREFERRED)
CHANNEL(C) QMNAME(CORE) CONNAME(core2.ops.company.example) CLNTWGHT(3) +
AFFINITY(PREFERRED)
CHANNEL(D) QMNAME(CORE) CONNAME(core3.ops.company.example) CLNTWGHT(2) +
AFFINITY(PREFERRED)
```

Problémy aplikace MQCONN (*CORE)

Kanál A není kandidátem pro toto připojení, protože atribut QMNAME se neshoduje. Kanály B, C a D jsou identifikovány jako kandidáti a jsou umístěny v pořadí předvolby na základě jejich váhy. V tomto příkladu by pořadí mohlo být C, B, D. Klient se pokusí připojit ke správci front v adresáři core2.ops.company.example. Název správce front na této adrese se nekontroluje, protože volání MQCONN obsahovalo v názvu správce front hvězdičku.

Je důležité si uvědomit, že při každém připojení tohoto konkrétního klientského počítače k produktu AFFINITY(PREFERRED) dojde k umístění kanálů ve stejném počátečním pořadí předvoleb. To platí i v případě, že připojení pocházejí z různých procesů nebo v různou dobu.

V tomto příkladu není možné dosáhnout správce front v souboru core.2.ops.company.example . Klient se pokusí připojit k souboru core1.ops.company.example , protože kanál B je další v pořadí preferované pořadí. Kromě toho je kanál C degradován, aby se stal nejméně preferovaným.

Druhé volání MQCONN (*CORE) je vydáno stejnou aplikací. Kanál C byl degradován předchozím připojením, takže nejpreferovanější kanál je nyní B. Toto připojení je vytvořeno pro core1.ops.company.example.

Druhý počítač, který sdílí stejnou tabulku definic kanálů klienta, umístí kanály do jiného výchozího pořadí předvolby. Například D, B, C. Za normálních okolností se všemi pracujícími kanály jsou aplikace na tomto počítači připojeny k souboru core3.ops.company.example , zatímco jsou na prvním počítači připojeny k souboru core2.ops.company.example. To umožňuje vyrovnávání pracovní zátěže u velkého počtu klientů ve více správcích front a zároveň umožnit každému jednotlivému klientovi připojit se ke stejnému správci front, je-li k dispozici.

Příklad 2. Výběr kanálů, je-li parametr ConnectionAffinity nastaven na hodnotu NONE
Tento příklad ilustruje, jak produkt IBM MQ MQI client vybere kanál z tabulky CCDT, kde je parametr ConnectionAffinity nastaven na hodnotu NONE.

V tomto příkladu používá řada klientů tabulku CCDT (Client Channel Definition Table) poskytovanou správcem front. Nástroje CCDT zahrnují kanály připojení klienta s následujícími atributy (zobrazené pomocí syntaxe příkazu DEFINE CHANNEL):

```
CHANNEL(A) QMNAME(DEV) CONNAME(devqm.it.company.example)
CHANNEL(B) QMNAME(CORE) CONNAME(core1.ops.company.example) CLNTWGHT(5) +
AFFINITY(NONE)
CHANNEL(C) QMNAME(CORE) CONNAME(core2.ops.company.example) CLNTWGHT(3) +
AFFINITY(NONE)
```

```
CHANNEL(D) QMNAME(CORE) CONNAME(core3.ops.company.example) CLNTWGHT(2) +
AFFINITY(NONE)
```

Aplikace vydá MQCONN (*CORE). Stejně jako v předchozím příkladu se kanál A nebere v úvahu, protože se neshoduje s hodnotou QMNAME. Kanál B, C nebo D je vybrán na základě jejich váhy, s pravděpodobností 50%, 30% nebo 20%. V tomto příkladu může být zvolen kanál B. Neexistuje žádné trvalé pořadí preferované předvolby.

Je provedeno druhé volání MQCONN (*CORE). Opět platí, že je vybrán jeden ze tří použitelných kanálů se stejnou pravděpodobností. V tomto příkladu je zvolen kanál C. Nicméně core2.ops.company.example neodpovídá, takže je mezi zbývajícími kandidátskými kanály provedena jiná volba. Je vybrán kanál B a aplikace je připojena k souboru core1.ops.company.example.

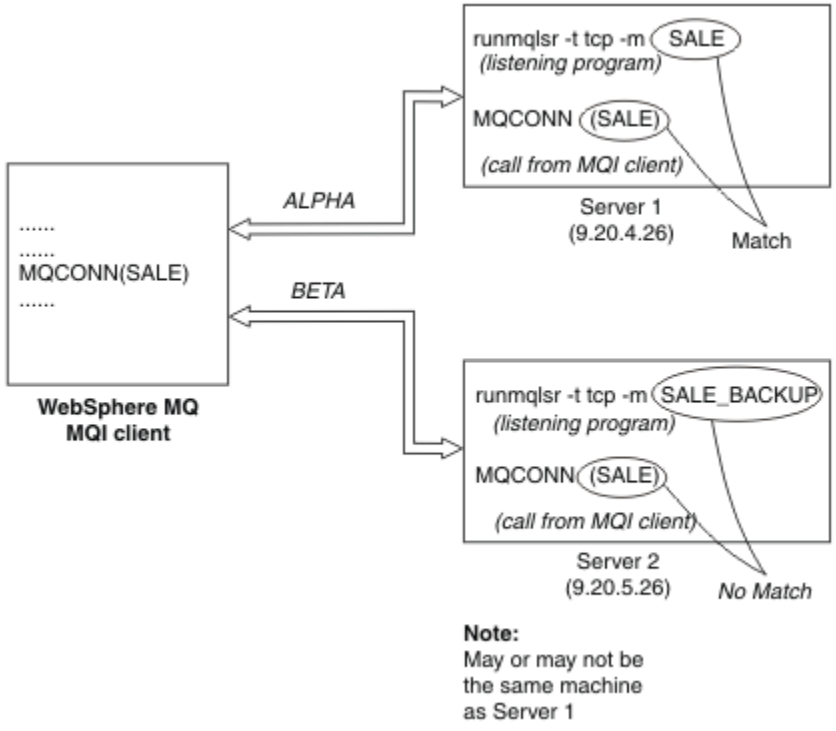
Při použití parametru AFFINITY (NONE) je každé volání MQCONN nezávislé na všech ostatních. Proto, když tato ukázková aplikace vytvoří třetí MQCONN (*CORE), může se pokusit o připojení prostřednictvím přerušeného kanálu C předtím, než jste vybrali jednu z hodnot B nebo D.

Příklady volání MQCONN

Příklady použití volání MQCONN pro připojení ke specifickému správci front nebo pro jednu ze skupin správců front.

V každém z následujících příkladů je síť stejná; existuje připojení definované pro dva servery ze stejného IBM MQ MQI client. (V těchto příkladech může být místo volání MQCONN použit volání MQCONNX.)

Na serverovém počítači běží dva správci front, jeden s názvem SALE a druhý s názvem SALE_BACKUP.



Obrázek 108. Příklad MQCONN

Definice pro kanály v těchto příkladech jsou:

Definice SALE:

```
DEFINE CHANNEL(ALPHA) CHLTYPE(SVRCONN) TRPTYPE(TCP) +
DESCR('Server connection to IBM MQ MQI client')

DEFINE CHANNEL(ALPHA) CHLTYPE(CLNTCONN) TRPTYPE(TCP) +
CONNAME(9.20.4.26) DESCR('IBM MQ MQI client connection to server 1') +
QMNAME(SALE)

DEFINE CHANNEL(BETA) CHLTYPE(CLNTCONN) TRPTYPE(TCP) +
CONNAME(9.20.5.26) DESCR('IBM MQ MQI client connection to server 2') +
QMNAME(SALE)
```

Definice SALE_BACKUP:

```
DEFINE CHANNEL(BETA) CHLTYPE(SVRCONN) TRPTYPE(TCP) +
DESCR('Server connection to IBM MQ MQI client')
```

Definice kanálů klienta lze shrnout takto:

Název	**CHLTYPE**	**TRPTYPE**	**CONNAME**	**QMNAME**
Alpha	CLNTCONN	TCP	9.20.4.26	PRODEJ
BETA	CLNTCONN	TCP	9.20.5.26	PRODEJ

Jaké příklady příkazů MQCONN demonstrují
Příklady demonstrují použití více správců front jako záložního systému.

Předpokládejme, že komunikační propojení k serveru 1 je dočasně přerušeno. Je předvedeno použití více správců front jako záložního systému.

Každý příklad pokrývá různé volání MQCONN a poskytuje vysvětlení toho, co se děje ve specifickém ukázkovém příkladu za použití následujících pravidel:

1. Tabulka CCDT (Client Channel Definition CCDT) je skenována v abecedním pořadí názvů kanálů pro název správce front (pole QMNAME) odpovídající danému názvu správce front zpráv v rámci volání MQCONN.
2. Je-li nalezena shoda, použije se definice kanálu.
3. Došlo k pokusu o spuštění kanálu na počítači identifikovaném názvem připojení (CONNAME). Je-li tato operace úspěšná, bude aplikace pokračovat. Vyžaduje:
   - Modul listener, který má být spuštěn na serveru.
   - Listener, který má být připojen ke stejnému správci front jako ten, ke kterému se klient chce připojit (je-li zadán).
4. Pokud se pokus o spuštění kanálu nezdaří a v tabulce definic kanálů klienta je více než jedna položka (v tomto příkladu jsou dvě položky), prohledá se soubor kvůli další shodě. Je-li nalezena shoda, zpracování pokračuje v kroku 1.
5. Pokud není nalezena žádná shoda nebo pokud v tabulce definic kanálů klienta nejsou žádné další položky a kanál se nepodařilo spustit, aplikace se nedokáže připojit. Ve volání MQCONN je vrácen příslušný kód příčiny a kód dokončení. Aplikace může provést akci na základě toho, jak byly vráceny kódy příčiny a dokončení.

Příklad 1. Název správce front obsahuje hvězdičku (*)
V tomto příkladu se aplikace nevztahuje na správce front, k němuž se připojuje. Aplikace vydá volání MQCONN pro název správce front včetně hvězdičky. Je zvolen vhodný kanál.

Problémy aplikace:

```
MQCONN (*SALE)
```

V souladu s pravidly se v této instanci děje toto:

1. Tabulka definic kanálů klienta (CCDT) je skenována pro název správce front SALE, odpovídá volání MQCONN aplikace.
2. Jsou nalezeny definice kanálů pro ALPHA a BETA .
3. Má-li jeden kanál hodnotu CLNTWGHT 0, je tento kanál vybrán. Pokud má obě hodnotu CLNTWGHT hodnotu 0, je kanál ALPHA vybrán, protože je první v abecedním pořadí. Mají-li oba kanály nenulová hodnota CLNTWGHT, je jeden kanál náhodně vybrán na základě jeho váhy.
4. Je proveden pokus o spuštění kanálu.
5. Byl-li vybrán kanál BETA , pokus o spuštění je úspěšný.
6. Pokud byl vybrán kanál ALPHA , pokus o jeho spuštění NEBYL úspěšný, protože komunikační spoj je poškozen. Pak se použijí následující kroky:
   a. Jediným dalším kanálem pro název správce front SALE je BETA.
   b. Pokus o spuštění tohoto kanálu je úspěšný-tento pokus je úspěšný.
7. Kontrola, zda je modul listener spuštěný, ukazuje, že je spuštěný jeden. Není připojen ke správci front produktu SALE , ale protože má parametr volání MQI v něm obsažený znak hvězdička (*), nekontroluje se žádná kontrola. Aplikace je připojena ke správci front produktu SALE_BACKUP a pokračuje ve zpracování.

Příklad 2. Zadán název správce front
V tomto příkladu se aplikace musí připojit ke konkrétnímu správci front. Aplikace vydá volání MQCONN pro daný název správce front. Je zvolen vhodný kanál.

Aplikace vyžaduje připojení ke specifickému správci front s názvem SALE, jak je uvedeno v rámci volání MQI:

```
MQCONN (SALE)
```

V souladu s pravidly se v této instanci děje toto:

1. Tabulka definic kanálů klienta (CCDT) je skenována v posloupnosti názvů abecedního kanálu, pro název správce front SALE, odpovídá volání MQCONN aplikace.
2. První nalezená definice kanálu je ALPHA.
3. Pokus o spuštění kanálu byl proveden-tento pokus není úspěšný, protože komunikační spoj je poškozen.
4. Byla znovu naskenována tabulka definic kanálů klienta pro název správce front SALE a byl nalezen název kanálu BETA .
5. Pokus o spuštění kanálu byl proveden-došlo k úspěšnému pokusu o spuštění kanálu.
6. Kontrola, zda modul listener běží, ukazuje, že je zde jeden spuštěný, ale není připojen ke správci front produktu SALE .
7. V tabulce definic kanálů klienta nejsou žádné další položky. Aplikace nemůže pokračovat a přijímá návratový kód MQRC_Q_MGR_NOT_AVAILABLE.

Příklad 3. Název správce front je prázdný nebo hvězdička (*)
V tomto příkladu se aplikace nevztahuje na správce front, k němuž se připojuje. Aplikace vydá příkaz MQCONN s uvedením prázdného názvu správce front nebo hvězdičky. Je zvolen vhodný kanál.

To je zpracovkládáno stejným způsobem jako “Příklad 1. Název správce front obsahuje hvězdičku (*)” na stránce 891.

Poznámka: Pokud byla tato aplikace spuštěna v jiném prostředí než IBM MQ MQI clienta název byl prázdný, pokus o připojení k výchozímu správci front se bude pokoušet připojit. To není případ, kdy je spuštěn z prostředí klienta. Přístup ke správci front je ten, který je přidružen k modulu listener, ke kterému se kanál připojuje.

Problémy aplikace:

```
MQCONN ("")
```

, nebo

```
MQCONN (*)
```

V souladu s pravidly se v této instanci děje toto:

1. Tabulka definic kanálů klienta (CCDT) je skenována v posloupnosti názvů abecedního kanálu, pro název správce front, který je prázdný, odpovídá volání MQCONN aplikace.
2. Položka pro název kanálu ALPHA má název správce front v definici SALE. Tato hodnota neodpovídá parametru volání MQCONN, který vyžaduje, aby název správce front byl prázdný.
3. Další položka je pro název kanálu BETA.
4. Definice `queue manager name` v definici je SALE. Tento parametr se znovu neshoduje s parametrem volání MQCONN, který vyžaduje, aby název správce front byl prázdný.
5. V tabulce definic kanálů klienta nejsou žádné další položky. Aplikace nemůže pokračovat a přijímá návratový kód MQRC_Q_MGR_NOT_AVAILABLE.

Spouštění v prostředí klienta

Zprávy odeslané aplikacemi produktu IBM MQ spuštěnými v produktu IBM MQ MQI clients přispívají ke spouštění stejným způsobem jako jiné zprávy a mohou být použity ke spouštění programů na serveru i na straně klienta.

Spouštěcí impuls je podrobně vysvětlen v části Spouštěcí kanály.

Monitor spouštěčů a aplikace, které mají být spuštěny, musí být na stejném systému.

Výchozí charakteristiky spuštěné fronty jsou stejné jako výchozí charakteristiky v prostředí serveru. Zejména pokud v aplikaci klienta při vkládání zpráv do spuštěné fronty, která je lokální vzhledem ke správci front produktu z/OS , nejsou určeny žádné volby řízení synchronizačního bodu MQPMO, zprávy jsou vloženy do jednotky práce. Je-li podmínka spouštěče splněna, zpráva spouštěče je vložena do inicializační fronty v rámci stejné jednotky práce a nemůže ji načíst monitor spouštěčů, dokud nebude ukončena jednotka práce. Proces, který se má spustit, se nespustí, dokud jednotka práce neskončí.

Definice procesu

Definici procesu musíte definovat na serveru, protože je to přidruženo k frontě, na které se spouští spouštěcí sada.

Objekt procesu definuje, co se má spustit. Pokud klient a server nejsou spuštění na stejné platformě, musí všechny procesy spuštěné monitorem spouštěčů definovat *ApplType*, jinak server převezme své výchozí definice (tj. typ aplikace, která je obvykle přidružena k počítači serveru) a způsobí selhání.

Je-li například monitor spouštěčů spuštěn na klientu Windows a chce odeslat požadavek na server v jiném operačním systému, musí být MQAT_WINDOWS_NT definován jinak, jinak jiný operační systém používá své výchozí definice a proces selže.

monitor spouštěčů

Monitor spouštěčů poskytovaný produkty jiného typu nežz/OS IBM MQ se spouští v prostředí klienta pro systémy IBM i IBM i, UNIX, Linux, and Windows .

Chcete-li spustit monitor spouštěčů, zadejte jeden z těchto příkazů:

- IBM i V systému IBM i:

```
CALL PGM(QMQM/RUNMQTMC) PARM('-m' QmgrName '-q' InitQ)
```

- ULW Na platformách Windows, UNIX and Linux :

```
runmqtmc [-m QMgrName] [-q InitQ]
```

Výchozí inicializační fronta je SYSTEM.DEFAULT.INITIATION.QUEUE ve výchozím správci front. Inicializační fronta je tam, kde monitor spouštěčů vyhledává zprávy spouštěče. Pak volá programy pro odpovídající zprávy spouštěče. Tento monitor spouštěčů podporuje výchozí typ aplikace a je stejný jako `runmqtrm` , až na to, že spojuje knihovny klienta.

Příkazový řetězec sestavený monitorem spouštěčů je následující:

1. *ApplicId* z příslušné definice procesu. *ApplicId* je název programu, který má být spuštěn, tak jak by byl zadán na příkazovém řádku.
2. Struktura MQTMC2 je uzavřena v uvozovkách, která byla získána z inicializační fronty. Je spuštěn příkazový řetězec, který má tento řetězec přesně tak, jak je zadán, v uvozovkách, aby jej příkaz systému přijal jako jeden parametr.
3. *EnvrData* z příslušné definice procesu.

Monitor spouštěčů se nepodívá, zda se v inicializační frontě nachází jiná zpráva, dokud není dokončeno spuštění aplikace. Pokud má aplikace příliš mnoho práce, monitor spouštěčů nemusí držet krok s počtem příchozích zpráv, které přicházejí do styku. Existují dva způsoby, jak se s touto situací vypořádat:

1. Mají spuštěné více monitorů spouštěčů

   Rozhodnete-li se spustit více monitorů spouštěčů, můžete řídit maximální počet aplikací, které mohou být spuštěny v libovolném okamžiku.
2. Spustit spuštěné aplikace na pozadí

   Zvolíte-li spuštění aplikací na pozadí, produkt IBM MQ nezavede žádné omezení počtu aplikací, které lze spustit.

Chcete-li spustit spuštěnou aplikaci na pozadí v systémech UNIX and Linux , musíte na konec definice procesu *EnvrData* vložit znak & (ampersand).

Aplikace produktu CICS (jiné nežz/OS)
Aplikační program non-z/OS CICS , který vydává volání MQCONN nebo MQCONNX , musí být definován jako CEDA jako RESIDENT. Pokud jako klienta znovu propojete aplikaci serveru CICS , riskujete ztrátu podpory synchronizačního bodu.

Aplikační program non-z/OS CICS , který vydává volání MQCONN nebo MQCONNX , musí být definován jako CEDA jako RESIDENT. Chcete-li, aby byl rezidentní kód co nejmenší, můžete vytvořit odkaz na samostatný program, který vydá volání MQCONN nebo MQCONNX .

Je-li proměnná prostředí MQSERVER použita k definování připojení klienta, musí být určena v parametru CICSENV.CMD .

Aplikace produktu IBM MQ lze spouštět v prostředí serveru IBM MQ nebo v klientu produktu IBM MQ bez změny kódu. V prostředí serveru IBM MQ však může produkt CICS vystupovat jako koordinátor synchronizačních bodů a vy používáte příkaz EXEC CICS SYNCPOINT a EXEC CICS SYNCPOINT ROLLBACK, nikoli **MQCMIT** a **MQBACK**. Je-li aplikace CICS jednoduše předána jako klient, podpora synchronizačních bodů se ztratí. **MQCMIT** a **MQBACK** musí být použity pro aplikaci spuštěnou na serveru IBM MQ MQI client.

Windows UNIX Příprava a spuštění aplikací CICS a Tuxedo

Chcete-li spustit produkt CICS a aplikace Tuxedo jako klientské aplikace, použijte různé knihovny od těch, které používáte s aplikacemi serveru. ID uživatele, pod kterým je aplikace spuštěna, je také odlišné.

Chcete-li připravit produkt CICS a aplikace Tuxedo, aby se spouštěli jako aplikace produktu IBM MQ MQI client , postupujte podle pokynů v tématu Konfigurace rozšířeného klienta transakcí.

Všimněte si však, že informace, které se zabývají speciálně připravováním aplikací CICS a Tuxedo, včetně ukázkových programů dodávaných s produktem IBM MQ, předpokládají, že připravujete aplikace ke spouštění na systému serveru IBM MQ . V důsledku toho se informace týkají pouze knihoven IBM MQ , které jsou určeny k použití na serverovém systému. Při přípravě aplikací klienta je třeba provést následující akce:

- Použijte příslušnou systémovou knihovnu klienta pro vazby jazyků, které vaše aplikace používá. Příklad:
  - UNIX Pro aplikace napsané v jazyce C v systému UNIXpoužijte knihovnu libmqic místo libmqm.
  - Windows V systémech Windows použijte místo mqm.libknihovnu mqic.lib .
- Místo systémových knihoven serveru, které jsou zobrazeny v Tabulka 119 na stránce 895 a Tabulka 120 na stránce 895, použijte ekvivalentní knihovny systému klienta. Není-li systémová knihovna serveru uvedena v těchto tabulkách, použijte stejnou knihovnu v klientském systému.

Tabulka 119. Systémové knihovny klienta v systému UNIX

Knihovna pro systém serveru IBM MQ	**Ekvivalentní knihovna pro použití na klientském systému IBM MQ**
libmqmxa	libmqcxa

Tabulka 120. Systémové knihovny klienta v systémech Windows

Knihovna pro systém serveru IBM MQ	**Ekvivalentní knihovna pro použití na klientském systému IBM MQ**
mqmxa.lib	mqcxa.lib
mqmtux.lib	mqcxa.lib
mqmenc.lib	mqcxa.lib
mqmcics4.lib	mqccics4.lib

ID uživatele použité klientskou aplikací

Když spustíte aplikaci serveru IBM MQ pod CICS, za normálních okolností se přepíná od uživatele CICS k ID uživatele transakce. Když však spustíte aplikaci IBM MQ MQI client pod CICS, zachovají si privilegované oprávnění CICS .

Windows UNIX *Ukázkové programy CICS a Tuxedo*

CICS a ukázkové programy produktu Tuxedo pro použití v systémech UNIX a Windows .

Tabulka 121 na stránce 895 uvádí ukázkové programy CICS a Tuxedo, které jsou dodávány pro použití v klientských systémech UNIX . Příkaz Tabulka 122 na stránce 896 vypíše ekvivalentní informace o klientských systémech Windows . Tabulky také zobrazují seznam souborů, které se používají pro přípravu a spuštění programů. Popis ukázkových programů viz “Ukázka transakce CICS” na stránce 1056 a “Použití ukázek TUXEDO na systémech UNIX a Windows” na stránce 1097.

Tabulka 121. Ukázkové programy pro klientské systémy UNIX

Popis	**Zdroj**	**Spustitelný modul**
CICS PROGRAM	amqscic0.ccs	amqscicc
Soubor záhlaví pro program CICS	amqscih0.h	-
Program klienta Tuxedo pro vložení zpráv	amqstxpx.c	-
Klientský program Tuxedo pro získání zpráv	amqstxgx.c	-
Serverový program Tuxedo pro dva klientské programy	amqstxsx.c	-

Tabulka 121. Ukázkové programy pro klientské systémy UNIX (pokračování)		
Popis	**Zdroj**	**Spustitelný modul**
Soubor UBBCONFIG pro programy Tuxedo	ubbstxcx.cfg	-
Soubor tabulky polí pro programy Tuxedo	amqstxvx.flds	-
Zobrazit soubor popisu pro programy Tuxedo	amqstxvx.v	-

Tabulka 122. Ukázkové programy pro klientské systémy Windows		
Popis	**Zdroj**	**Spustitelný modul**
CICS transakce	amqscic0.ccs	amqscicc
Soubor záhlaví pro transakci CICS	amqscih0.h	-
Program klienta Tuxedo pro vložení zpráv	amqstxpx.c	-
Klientský program Tuxedo pro získání zpráv	amqstxgx.c	-
Serverový program Tuxedo pro dva klientské programy	amqstxsx.c	-
Soubor UBBCONFIG pro programy Tuxedo	ubbstxcx.cfg	-
Soubor tabulky polí pro programy Tuxedo	amqstxvx.fld	-
Zobrazit soubor popisu pro programy Tuxedo	amqstxvx.v	-
Makefile pro programy Tuxedo	amqstxmc.mak	-
Soubor ENVFILE pro programy Tuxedo	amqstxen.env	-

Windows UNIX *Chybová zpráva: AMQ5203, jak je upraveno pro aplikace CICS a aplikace Tuxedo*

Spustíte-li produkt CICS nebo aplikace Tuxedo, které používají rozšířeného transakčního klienta, mohou se zobrazit standardní diagnostické zprávy. Jeden z nich byl upraven pro použití s rozšířeným transakčním klientem

Zprávy, které se mohou zobrazit v souborech protokolu chyb produktu IBM MQ , jsou zdokumentovány v tématu Diagnostické zprávy: AMQ4000-9999. Zpráva AMQ5203 byla upravena pro použití s rozšířeným transakčním klientem. Zde je text upravené zprávy:

AMQ5203: Došlo k chybě při volání rozhraní XA.

Vysvětlení

Číslo chyby je & 2, kde hodnota 1 označuje, že zadaná hodnota příznaků '& 1 byla neplatná, 2 označuje, že došlo k pokusu o použití podprocesů s podporou podprocesů a bez podprocesů ve stejném procesu, 3 označuje, že došlo k chybě s dodaným názvem správce front' & 3 ', 4 označuje, že ID správce prostředků & 1 je neplatné, 5 označuje, že byl proveden pokus o použití druhého správce front s názvem'& 3 'když byl jiný správce front již připojen, 6 označuje, že správce transakcí byl volán, když aplikace není připojena ke správci front, 7 označuje, že volání XA bylo provedeno během dalšího volání, 8 označuje, že řetězec xa_info' & 4 'v volání xa_open obsahoval neplatnou hodnotu parametru pro název parametru' & 5 ', a 9 označuje, že řetězec xa_info' & 4 'v volání xa_open postrádá požadovaný parametr, název parametru' & 5 '.

Odezva uživatele

Opravte chybu a zkuste operaci znovu.

Příprava a spuštění aplikací serveru Microsoft Transaction Server

Chcete-li připravit aplikaci MTS ke spuštění jako aplikaci produktu IBM MQ MQI client , postupujte podle těchto pokynů, jak je to vhodné pro vaše prostředí.

Obecné informace o tom, jak vyvíjet aplikace MTS (Microsoft Transaction Server), které přistupují k prostředkům produktu IBM MQ , najdete v části týkající se MTS v Centru nápovědy IBM MQ .

Chcete-li připravit aplikaci MTS ke spuštění jako aplikaci produktu IBM MQ MQI client , proveďte jednu z následujících komponent pro každou komponentu aplikace:

- Pokud komponenta používá vazby jazyka C pro rozhraní MQI, postupujte podle pokynů v části “Příprava programů jazyka C v produktu Windows” na stránce 994 , ale propojte ji s knihovnou mqicxa.lib místo mqic.lib.
- Pokud komponenta používá třídy C++ IBM MQ , postupujte podle pokynů v “Sestavování programů C++ v systému Windows” na stránce 510 , ale propojte komponentu s knihovnou imqx23vn.lib místo imqc23vn.lib.
- Pokud komponenta používá vazby jazyka Visual Basic pro rozhraní MQI, postupujte podle pokynů v příručce “Příprava programů produktu Visual Basic v produktu Windows” na stránce 998 , ale při definování projektu Visual Basic zadejte do pole **Argumenty podmíněné kompilace** hodnotu `MqType=3` .
- Pokud komponenta používá třídy automatizace IBM MQ pro ActiveX (MQAX), definujte proměnnou prostředí GMQ_MQ_LIB s hodnotou `mqic32xa.dll`.

  Proměnnou prostředí můžete definovat ve své aplikaci, nebo ji můžete definovat tak, aby její rozsah byl celý systém. Definováním v celém systému však může dojít k nesprávnému chování existující aplikace MQAX, která nedefinuje proměnnou prostředí v rámci aplikace.

Příprava a spuštění aplikací produktu IBM MQ JMS

Aplikace produktu IBM MQ JMS v režimu klienta můžete spouštět v produktu WebSphere Application Server jako správce transakcí. Mohou se zobrazit určité varovné zprávy.

Chcete-li připravit a spustit aplikace produktu IBM MQ JMS v režimu klienta pomocí produktu WebSphere Application Server jako svého správce transakcí, postupujte podle pokynů v části “použitíIBM MQ classes for JMS” na stránce 69.

Když spustíte klientskou aplikaci IBM MQ JMS , můžete zobrazit následující varovné zprávy:

MQJE080
Nedostatečné licenční jednotky-spusťte příkaz setmqcap

MQJE081
Soubor obsahující informace o licenční jednotce je ve špatném formátu-spusťte příkaz settqcap

MQJE082
Soubor obsahující informace o jednotce licence nebyl nalezen-spusťte příkaz settmqcap

Uživatelské procedury, uživatelské procedury rozhraní API a instalovatelné služby produktu IBM MQ

Toto téma obsahuje odkazy na informace o používání a vývoji těchto programů.

Úvod do způsobu použití uživatelských procedur, uživatelských procedur rozhraní API a instalovatelných služeb pro rozšíření zařízení správce front najdete v tématu Rozšíření zařízení správce front.

Informace o psaní a kompilaci uživatelských procedur a instalovatelných službách najdete v dílčích tématech.

Související informace

Programy pro ukončení kanálů pro kanály MQI

Popis uživatelské procedury rozhraní

Referenční informace o rozhraní instalovatelných služeb

IBM i Referenční informace o rozhraní instalovatelných služeb v systému IBM i

ULW Zapisování uživatelských procedur a instalovatelných služeb na UNIX, Linux a Windows

Můžete psát a kompilovat uživatelské procedury bez odkazů na libovolné knihovny produktu IBM MQ v systémech UNIX, Linux a Windows.

Informace o této úloze

Toto téma platí pouze pro systémy UNIX, Linux, and Windows . Podrobnosti o zápisu uživatelských procedur a instalovatelných služeb pro jiné platformy naleznete v příslušných tématech specifických pro platformu.

Je-li produkt IBM MQ nainstalován v jiném než výchozím umístění, musíte vytvořit a zkompilovat uživatelské procedury bez odkazování na žádné knihovny produktu IBM MQ .

V systémech UNIX, Linux, and Windows můžete zapisovat a kompilovat bez propojení jedné z těchto knihoven produktu IBM MQ :

- mqmzf
- MQM
- mqmvx
- mqmvxd
- mqiová
- mkvl

Existující uživatelské procedury, které jsou propojeny s těmito knihovnami, budou pokračovat v práci, takže je v systému UNIX and Linux nainstalován produkt IBM MQ ve výchozím umístění.

Postup

1. Zahrňte hlavičkový soubor cmqec.h .

   Začlenění tohoto souboru záhlaví automaticky zahrnuje soubory záhlaví cmqc.h, cmqxc.h a cmqzc.h .
2. Zadejte uživatelskou proceduru tak, aby byla volání MQI a DCI prováděna prostřednictvím struktury MQIEP. Další informace o struktuře MQIEP naleznete v tématu Struktura MQIEP.
   - Instalovatelné služby
     - Pomocí parametru **Hconfig** lze odkazovat na volání MQZEP.
     - Před použitím parametru **Hconfig** je třeba zkontrolovat, zda první 4 bajty **Hconfig** odpovídají struktuře **StrucId** struktury MQIEP.
     - Další informace o zápisu instalovatelných komponent služeb najdete v tématu MQIEP.
   - Uživatelské procedury rozhraní API
     - Pomocí parametru **Hconfig** lze odkazovat na volání MQXEP.
     - Před použitím parametru **Hconfig** je třeba zkontrolovat, zda první 4 bajty **Hconfig** odpovídají struktuře **StrucId** struktury MQIEP.
     - Další informace o zápisu uživatelských procedur rozhraní API najdete v tématu “Zápis uživatelských procedur API” na stránce 918.
   - Uživatelské procedury kanálu
     - Pomocí argumentu **pEntryPoints** struktury MQCXP lze odkazovat na volání MQI a DCI.
     - Před použitím produktu **pEntryPoints**je třeba zkontrolovat, zda je číslo verze MQCXP verze 8 nebo vyšší.
     - Další informace o zápisu uživatelských procedur kanálů naleznete v příručce “Psaní programů výstupních bodů kanálu” na stránce 929.
   - Ukončení převodu dat

- Použijte parametr **pEntryPoints** struktury MQDXP, aby ukazoval na volání MQI a DCI.
- Před použitím produktu **pEntryPoints**je třeba zkontrolovat, zda je číslo verze MQDXP verze 2 nebo vyšší.
- Můžete použít příkaz **crtmqcvx** a zdrojový soubor amqsvfc0.c k vytvoření kódu pro převod dat, který používá parametr **pEntryPoints** . Další informace jsou uvedeny v tématech "Psaní uživatelské procedury pro převod dat pro IBM MQ for Windows" na stránce 957 a "Psaní uživatelské procedury pro převod dat pro produkt IBM MQ v systémech UNIX and Linux" na stránce 953.
- Pokud máte existující uživatelské procedury pro převod dat, které byly vygenerovány pomocí příkazu **crtmqcvx** , je nutné ukončit uživatelskou proceduru pomocí aktualizovaného příkazu.
- Další informace o zápisu uživatelských procedur pro převod dat najdete v tématu "Zápis uživatelských procedur pro převod dat" na stránce 948.

- Uživatelské procedury před připojením
  - Pomocí argumentu **pEntryPoints** struktury MQNXP lze odkazovat na volání MQI a DCI.
  - Před použitím produktu **pEntryPoints**je třeba zkontrolovat, zda je číslo verze MQNXP verze 2 nebo vyšší.
  - Další informace o zápisu východů před připojením naleznete v tématu "Odkazování na definice připojení pomocí předání před připojením z úložiště" na stránce 959.
- Uživatelské procedury publikování
  - Pomocí argumentu **pEntryPoints** struktury MQPSXP lze odkazovat na volání MQI a DCI.
  - Před použitím produktu **pEntryPoints**je třeba zkontrolovat, zda je verze MQPXP verze ve verzi 2 nebo vyšší.
  - Další informace o zápisu uživatelských procedur pro publikování naleznete v tématu "Zápis a kompilace uživatelských procedur pro publikování" na stránce 961.
- Ukončení pracovní zátěže klastru
  - Pomocí argumentu **pEntryPoints** struktury MQWXP lze odkazovat na volání MQXCLWLN.
  - Před použitím produktu **pEntryPoints**je třeba zkontrolovat, zda je verze MQWXP verze 4 nebo vyšší.
  - Další informace o zápisu uživatelských procedur pracovní zátěže klastru najdete v tématu "Zápis a kompilace uživatelských procedur pracovní zátěže klastru" na stránce 962.

Například v uživatelské proceduře kanálu volání MQPUT postupujte takto:

```
pChannelExitParms -> pEntryPoints -> MQPUT_Call(pChannelExitParms -> Hconn,
                                                Hobj,
                                                &md,
                                                &pmo,
                                                messlen,
                                                buffer,
                                                &CompCode,
                                                &Reason);
```

Další příklady lze zobrazit v příručce "Použití ukázkových procedurálních programů produktu IBM MQ" na stránce 1034.

3. Zkompilujte uživatelskou proceduru:
   - Nepropojujte s knihovnami produktu IBM MQ .
   - Nezahrnujte vestavěnou cestu RPath do žádných knihoven produktu IBM MQ ve vaší uživatelské proceduře.
   - Další informace o kompilaci uživatelské procedury naleznete v jednom z následujících témat:
     - Uživatelské procedury rozhraní API: "Kompilace uživatelských procedur rozhraní API" na stránce 919.

- Uživatelské procedury kanálu, uživatelské procedury publikování, ukončení pracovní zátěže klastru: “Kompilace ukončovacích programů kanálu v systémech Windows, UNIX and Linux” na stránce 947.
- Uživatelské procedury pro převod dat: “Zápis uživatelských procedur pro převod dat” na stránce 948.

4. Oďte uživatelskou proceduru v jednom z následujících míst:
   - Cesta k vašemu výběru při konfiguraci uživatelské procedury
   - Výchozí cesta k ukončení, ve specifickém instalačním adresáři. Například *MQ_DATA_PATH*/exits/installation2.
   - Výchozí cesta k ukončení

     Výchozí cesta k výstupní cestě je *MQ_DATA_PATH*/exits pro 32 bitových východů a *MQ_DATA_PATH*/exits64 pro 64bitové procedury. Tyto cesty můžete změnit v souboru qm.ini nebo mqclient.ini . Další informace najdete v tématu Cesta k uživatelské proceduře. V systémech Windows a Linuxmůžete pro změnu cesty použít Průzkumníka IBM MQ :

     a. Klepněte pravým tlačítkem myši na název správce front.
     b. Klepněte na **Vlastnosti ...**
     c. Klepněte na **Uživatelské procedury**
     d. Do pole výchozí cesty východů zadejte cestu k adresáři, který obsahuje ukončovací program.

   Je-li uživatelská procedura umístěna do specifického instalačního adresáře a do výchozího adresáře cesty, použije se k instalaci produktu IBM MQ v cestě k dispozici specifický instalační adresář instalačního adresáře. Například, výstup je umístěn v /exits/installation2 a v /exits, ale ne v /exits/installation1. Instalace produktu IBM MQ installation2 používá ukončení programu /exits/installation2. Instalace produktu IBM MQ installation1 používá ukončení z adresáře /exits .
5. Je-li to nezbytné, nakonfigurujte uživatelskou proceduru:
   - Instalovatelné služby: “Konfigurace služeb a komponent” na stránce 908.
   - Uživatelské procedury rozhraní API: “Konfigurace uživatelských procedur rozhraní API” na stránce 923.
   - Uživatelské procedury kanálu: “Konfigurace uživatelských procedur kanálu” na stránce 948.
   - Uživatelské procedury publikování: “Konfigurace uživatelských procedur publikování” na stránce 962.
   - Předběžné připojení se ukončí: stanzaPreConnect konfiguračního souboru klienta.

ULW *Uživatelské procedury rozhraní API nejsou propojeny s knihovnou MQI*

Za určitých okolností byste měli propojit existující uživatelskou proceduru rozhraní API, kterou nelze znovu zakódovat pro použití ukazatelů funkcí MQIEP s knihovnou rozhraní API produktu IBM MQ .

To je nezbytné, aby bylo možné úspěšně zavést existující uživatelskou proceduru rozhraní API za běhu programu linker vašeho systému do programů, které ještě nemají naložené ukazatele funkcí.

Poznámka: Tyto informace jsou omezeny na ty existující uživatelské procedury rozhraní API, které provádí volání MQI přímo. To znamená, že tyto uživatelské procedury se nepoužívají, MQIEP. Je-li to možné, měli byste namísto toho naplánovat změnu kódu pro ukončení použití vstupních bodů MQIEP.

V produktu IBM MQ 8.0je produkt **runmqsc** příkladem programu, který se přímo nespojí s knihovnou MQI.

Proto uživatelská procedura rozhraní API, která nebyla propojena se svou vyžadovanou knihovnou rozhraní API produktu IBM MQ nebo překódovaná pro použití rozhraní MQIEP, se nepodařilo načíst do produktu **runmqsc**.

Zobrazí se chyby v protokolu chyb správce front, například AMQ6175: Systém nemohl dynamicky načíst sdílenou knihovnu, společně s kvalifikačním textem, jako je například produkt undefined symbol: MQCONN.

a AMQ7214: `Modul pro uživatelskou proceduru rozhraní API 'myexitname' nelze načíst.`

Související úlohy

“Zapisování uživatelských procedur a instalovatelných služeb na UNIX, Linux a Windows” na stránce 898
Můžete psát a kompilovat uživatelské procedury bez odkazů na libovolné knihovny produktu IBM MQ v systémech UNIX, Linux a Windows.

ULW Instalovatelné služby a komponenty pro produkty UNIX, Linux a Windows

Tento oddíl uvádí instalovatelné služby a funkce a komponenty, které jsou k nim přidruženy. Rozhraní pro tyto funkce je dokumentováno tak, že vy nebo dodavatelé softwaru můžete dodávat komponenty.

Hlavní důvody pro poskytování instalovatelných služeb produktu IBM MQ jsou:

- Chcete-li vám poskytnout flexibilitu při výběru toho, zda mají být použity komponenty poskytované produkty IBM MQ , nebo je můžete nahradit nebo je rozšiřovat s ostatními.
- Chcete-li dodavatelům umožnit účast, tím, že poskytnete komponenty, které mohou využívat nové technologie, aniž byste provedli interní změny v produktech IBM MQ .
- Umožněte produktu IBM MQ využívat rychlejší a levnější využití nových technologií, a proto poskytovat produkty dříve a za nižší ceny.

Instalovatelné služby a *komponenty služeb* jsou součástí struktury produktu IBM MQ . Ve středu této struktury je ta část správce front, která implementuje danou funkci a pravidla přidružená k rozhraní MQI (Message Queue Interface). Tato centrální část vyžaduje řadu servisních funkcí nazývaných *instalovatelné služby*, aby mohla provést svou práci. Instalovatelné služby jsou:

- Autorizační služba
- služba názvů

Každá instalovatelná služba je související sadou funkcí implementovaných pomocí jedné nebo více *komponent služeb*. Každá komponenta je vyvolána pomocí veřejně dostupného rozhraní, které je k dispozici. To umožňuje nezávislým dodavatelům softwaru a dalším třetím stranám poskytnout instalovatelné komponenty k rozšíření nebo nahrazení produktů poskytovaných produkty IBM MQ . Tabulka 123 na stránce 901 shrnuje služby a komponenty, které lze použít.

Tabulka 123. Souhrn komponent instalovatelné služby

instalovatelná služba	**Dodaná komponenta**	**Funkce**	**Požadavky**
Autorizační služba	správce oprávnění k objektu (OAM)	Poskytuje kontrolu autorizace pro příkazy a volání MQI. Uživatelé mohou napsat svou vlastní komponentu pro rozšíření nebo nahrazení OAM. Chcete-li například zkontrolovat, zda má ID uživatele oprávnění k otevření fronty.	(Předpokládá se vhodná oprávnění k autorizaci platformy)
služba názvů	Není	Poskytuje podporu pro správce front za účelem vyhledání názvu správce front, který vlastní určenou frontu. • Definované uživatelem	• třetí strana nebo uživatel zapsaný jménem uživatele

Rozhraní instalovatelných služeb je popsáno v tématu Referenční informace o rozhraní instalovatelných služeb.

Související informace

Konfigurace instalovatelných služeb

Psaní komponenty služby

Tato sekce popisuje vztah mezi službami, komponentami, vstupními body a návraty k návratovému kódu.

Funkce a komponenty

Každá služba se skládá ze sady souvisejících funkcí. Např. služba názvů obsahuje funkci pro:

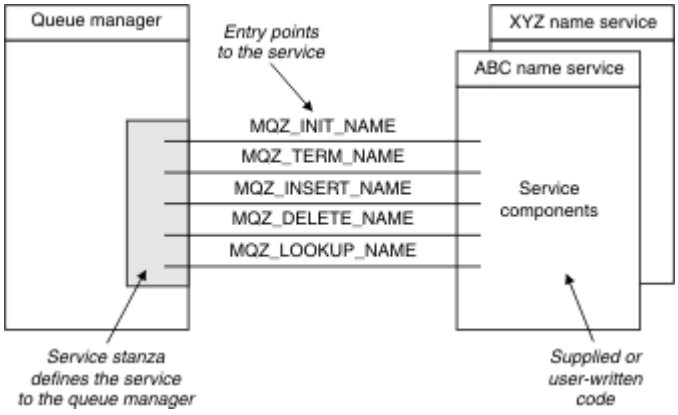
- Probíhá hledání názvu fronty a vrácení názvu správce front, ve kterém je fronta definována.
- Vložení názvu fronty do adresáře služby
- Odstranění názvu fronty z adresáře služby

Obsahuje také funkce inicializace a ukončení.

Instalovatelná služba je poskytována jednou nebo více komponentami služeb. Každá komponenta může provádět některé nebo všechny funkce, které jsou pro danou službu definovány. Například v produktu IBM MQ for AIXposkytuje dodaná komponenta autorizační služby, OAM, všechny dostupné funkce. Další informace viz “Rozhraní autorizační služby” na stránce 905. Komponenta je také odpovědná za správu veškerých podkladových prostředků nebo softwaru (například adresáře LDAP), které potřebuje k implementaci služby. Konfigurační soubory poskytují standardní způsob, jak načíst komponentu a určit adresy funkčních rutin, které poskytuje.

Produkt Obrázek 109 na stránce 902 ukazuje, jak spolu souvisí služby a komponenty:

- Služba je definována pro správce front podle oddílů v konfiguračním souboru.
- Každá služba je podporována zadaným kódem ve správci front. Uživatelé nemohou tento kód změnit, a proto nemohou vytvořit své vlastní služby.
- Každá služba je implementována jednou nebo více komponentami. Tyto služby mohou být dodány spolu s produktem nebo uživatelem napsanými. Je možné vyvolat více komponent pro službu, přičemž každá z nich podporuje různá zařízení v rámci služby.
- Vstupní body spojují komponenty služeb s podpůrným kódem ve správci front.



Obrázek 109. Základní informace o službách, komponentách a vstupních bodech

Vstupní body

Každá komponenta služby je představována pomocí seznamu adres vstupního bodu rutin, které podporují konkrétní instalovatelnou službu. Instalovatelná služba definuje funkci, která má být prováděna každou rutinou.

Uspořádání komponent služeb, když jsou nakonfigurovány, definuje pořadí, ve kterém jsou vstupní body volány ve snaze vyhovět požadavku na službu.

V dodaném hlavičkovém souboru `cmqzc.h` mají dodávané vstupní body pro každou službu předponu MQZID_.

Jsou-li služby přítomny, služby se načtou v předdefinovaném pořadí. Následující seznam zobrazuje služby a pořadí, ve kterém jsou inicializovány.

1. `NameService`
2. `AuthorizationService`
3. `UserIdentifierService`

`AuthorizationService` je jediná služba, která je ve výchozím nastavení konfigurována. Pokud je chcete použít, konfigurujte produkty `NameService` a `UserIdentifierService` ručně.

Služby a komponenty služeb mají mapování jeden-na-jednoho nebo jeden-na-mnoho. Pro každou službu lze definovat více komponent služby. Na systémech UNIX and Linux se hodnota Service Service stanza `ServiceComponent` musí shodovat s hodnotou názvu servisní sekce v souboru qm.ini . V systému Windowsmusí hodnota klíče registru služeb produktu `ServiceComponent` odpovídat hodnotě klíče registru názvu a je definována jako: `HKEY_LOCAL_MACHINE\SOFTWARE\IBM\WebSphere MQ\Installation\`*`MQ_INSTALLATION_NAME`*`\Configuration\QueueManager\`*`qmname`*`\` , kde *qmname* je název správce front.

Pro systémy UNIX and Linux se komponenty služeb spouštějí v pořadí, ve kterém jsou definovány v souboru qm.ini . V systému Windows, protože je použit registr Windows , IBM MQ vydává volání **RegEnumKey** , které vrací hodnoty v abecedním pořadí. Proto se při Windows služby volají v abecedním pořadí, jak jsou definovány v registru.

Pořadí definic `ServiceComponent` je významné. Toto pořadí určuje pořadí, ve kterém jsou komponenty spuštěny pro danou službu. Například, `AuthorizationService` na Windows je nakonfigurovaný s výchozí komponentou OAM s názvem `MQSeries.WindowsNT.auth.service`. Další komponenty lze definovat pro tuto službu, aby bylo možné přepsat výchozí hodnotu OAM. Pokud není zadán parametr `MQCACF_SERVICE_COMPONENT` , použije se první komponenta rozpoznala v abecedním pořadí ke zpracování požadavku a použije se název této komponenty.

Návratové kódy

Komponenty služeb poskytují správci front návratové kódy pro vytváření sestav o různých podmínkách. Ohlašují úspěch nebo selhání operace a označují, zda má správce front pokračovat do další komponenty služby. Samostatná hodnota parametru *Continuation* je tato indikace.

Data komponent

Jedna komponenta služby může vyžadovat sdílení dat mezi různými funkcemi. Instalovatelné služby poskytují volitelnou datovou oblast, která má být předána při každém vyvolání komponenty služby. Tato datová oblast je určena pro výlučné použití komponenty služby. Je sdílen všemi vyvoláními určité funkce, i když jsou prováděna z různých adresních prostorů nebo procesů. Je zaručeno, že bude adresovatelný z komponenty služby, kdykoli se zavolá. Je třeba deklarovat velikost této oblasti ve stanze *`ServiceComponent`* .

Inicializace a ukončení komponent
Použití voleb inicializace a ukončení komponenty.

Při vyvolání rutiny inicializace komponenty musí být volána funkce **MQZEP** správce front pro každý vstupní bod podporovaný danou komponentou. **MQZEP** definuje vstupní bod do služby. Předpokládá se, že všechny nedefinované výstupní body jsou NULL.

Komponenta se vždy vyvolá jednou s primární volbou inicializace, dříve než je vyvolána jiným způsobem.

Komponenta může být vyvolána se sekundární volbou inicializace na určitých platformách. Může být například vyvolán jednou pro každý proces operačního systému, podproces nebo úlohu, ke které je služba přistupována.

Je-li použita sekundární inicializace:

- Komponenta může být vyvolána více než jednou pro sekundární inicializaci. Pro každé takové volání se vydá odpovídající volání pro sekundární ukončení, když již služba není potřebná.

  Pro služby názvů se jedná o volání MQZ_TERM_NAME.

  U autorizačních služeb se jedná o volání MQZ_TERM_AUTHORITY.
- Vstupní body musí být znovu zadány (voláním MQZEP) pokaždé, když je komponenta volána pro primární a sekundární inicializaci.
- Pro komponentu se použije pouze jedna kopie dat komponenty; pro každou sekundární inicializaci není jiná kopie.
- Komponenta není vyvolána pro žádné další volání na službu (z procesu operačního systému, vlákna nebo úlohy, jak je to vhodné) před sekundární inicializací.
- Komponenta musí nastavit parametr **Version** na stejnou hodnotu pro primární a sekundární inicializaci.

Komponenta se vždy vyvolá s jednou volbou primárního ukončení jednou, když již není potřeba. K této komponentě nejsou vytvořena žádná další volání.

Komponenta je vyvolána se sekundární volbou ukončení, pokud byla vyvolána pro sekundární inicializaci.

správce oprávnění k objektu (OAM)
Komponenta autorizační služba dodaná s produkty IBM MQ se nazývá OAM (Object Authority Manager).

Ve výchozím nastavení je OAM aktivní a pracuje s řídicími příkazy **dspmqaut** (oprávnění k zobrazení), **dmpmqaut** (oprávnění k výpisu) a **setmqaut** (sada nebo reset oprávnění).

Syntaxe těchto příkazů a jejich použití jsou popsány v Referenční příručce řídicích příkazů produktuIBM MQ.

OAM pracuje s *entitou* činitele nebo skupiny:

- Linux UNIX V systémech UNIX and Linux je činitel ID uživatele nebo ID přidružené k aplikačnímu programu spuštěnému jménem uživatele; tato skupina je systémem definovaná kolekce činitelů.
- Windows V systému Windows je činitel ID uživatele produktu Windows nebo ID přidružené k aplikačnímu programu spuštěnému jménem uživatele; skupina je skupinou Windows .

Oprávnění mohou být udělována nebo odvolány na úrovni činitele nebo skupiny.

Když je vydán požadavek MQI nebo je vydán příkaz, OAM zkontroluje, zda má entita přidružená k operaci autorizaci k provedení požadované operace a k přístupu k uvedeným prostředkům správce front.

Autorizační služba vám umožňuje rozšířit nebo nahradit kontrolu oprávnění poskytovanou pro správce front tím, že zapisujete svou vlastní komponentu autorizační služby.

služba názvů
Služba názvů je instalovatelná služba, která poskytuje podporu správci front za účelem vyhledání názvu správce front, který vlastní uvedenou frontu. Z názvu služby nelze načíst žádné jiné atributy fronty.

Služba názvů umožňuje aplikaci otevřít vzdálenou frontu pro výstup tak, jako by šlo o lokální fronty. Služba názvů není vyvolána pro objekty jiné než fronty.

Poznámka: Vzdálené fronty musí mít nastaven atribut **Scope** nastaven na CELL.

Když aplikace otevře frontu, vyhledá nejprve název fronty v adresáři správce front. Pokud ji nenajde, prohledá tolik služeb názvů, kolik bylo nakonfigurováno, dokud nenajde takový, který rozpoznává název fronty. Pokud žádný z nich nerozpozná jméno, otevře se otevření.

Služba názvů vrací vlastníka správce front pro danou frontu. Správce front bude poté pokračovat s požadavkem MQOPEN, jako kdyby příkaz určil název fronty a správce front v původní žádosti.

Rozhraní NSI (name service interface) je součástí rámce IBM MQ .

Jak funguje služba názvů

Pokud definice fronty určuje atribut **Scope** jako správce front, tj. SCOPE (QMGR) v prostředí MQSC, je definice fronty (spolu se všemi atributy fronty) uložena pouze v adresáři správce front. Tuto volbu nelze nahradit instalovatelnou službou.

Pokud definice fronty uvádí atribut **Scope** jako buňku, tj. SCOPE (CELL) v prostředí MQSC, je definice fronty znovu uložena v adresáři správce front spolu se všemi atributy fronty. Název fronty a správce front je však také uložen ve službě názvů. Není-li k dispozici žádná služba, která by mohla tyto informace uložit, nelze definovat frontu s buňkou *Scope* .

Adresář, ve kterém jsou informace uloženy, může být spravován službou, nebo může služba použít základní službu, například adresář LDAP pro tento účel. V obou případech musí být definice uložené v adresáři zachovány i po ukončení komponenty a správce front, dokud nejsou explicitně odstraněny.

Poznámka:

1. Chcete-li odeslat zprávu do definice lokální fronty vzdáleného hostitele (s rozsahem CELL) na jiném správci front v rámci buňky adresáře pojmenovávání, je třeba definovat kanál.
2. Zprávy nelze načíst přímo ze vzdálené fronty, a to ani v případě, že má rozsah CELL.
3. Při odesílání do fronty s rozsahem CELL není vyžadována žádná definice vzdálené fronty.
4. Služba názvů centrálně definuje cílovou frontu, ačkoli stále ještě potřebujete přenosovou frontu k cílovému správci front a dvojici definic kanálů. Kromě toho musí mít přenosová fronta v lokálním systému stejný název jako správce front, který vlastní cílovou frontu, s oborem buňky, ve vzdáleném systému.

   Pokud má například vzdálený správce front název QM01, přenosová fronta v lokálním systému musí mít také název QM01.

Rozhraní autorizační služby
Autorizační služba poskytuje vstupní body pro použití správcem front.

Vstupní body jsou následující:

MQZ_AUTHENTICATE_USER
Ověřuje ID uživatele a heslo a může nastavit pole kontextu identity.

MQZ_CHECK_AUTHORITY
Kontroluje, zda má entita oprávnění provést jednu nebo více operací na uvedeném objektu.

MQZ_CHECK_PRIVILEGED
Kontroluje, zda je určený uživatel privilegovaným uživatelem.

MQZ_COPY_ALL_AUTHORITY
Kopíruje všechna aktuální oprávnění, která existují pro odkazovaný objekt, na jiný objekt.

OPRÁVNĚNÍ MQZ_DELETE_AUTHORITY
Odstraní všechny autorizace přidružené k uvedenému objektu.

MQZ_ENUMERATE_AUTHORITY_DATA
Načte všechna data oprávnění, která se shodují s uvedenými kritérii výběru.

MQZ_FREE_USER
Uvolní přidružené přidělené prostředky.

FUNKCE MQZ_GET_AUTHORITY
Získá oprávnění, které má entita pro přístup k uvedenému objektu.

MQZ_GET_EXPLICITNÍ_AUTORITA
Získá buď oprávnění, které má pojmenovaná skupina k přístupu k uvedenému objektu (ale bez dalšího oprávnění skupiny **nikdo**), nebo oprávnění, které má primární skupina uvedeného hlavního objektu k přístupu k uvedenému objektu.

MQZ_INIT_AUTHORITY
Inicializuje komponentu autorizační služby.

MQZ_DOTÁZAT SE
Dotáže se podporované funkčnosti autorizační služby.

MQZ_REFRESH_CACHE
Aktualizujte všechny autorizace.

OPRÁVNĚNÍ MQZ_SET_AUTHORITY
Nastaví oprávnění, které má entita k uvedenému objektu.

OPRÁVNĚNÍ MQZ_TERM_AUTHORITY
Ukončí komponentu autorizační služby.

Kromě toho v produktu IBM MQ for Windowsposkytuje autorizační služba následující vstupní body určené pro použití správcem front:

- **MQZ_CHECK_AUTHORITY_2**
- **MQZ_GET_AUTHORITY_2**
- **MQZ_GET_EXPLICIT_AUTHORITY_2**
- **MQZ_SET_AUTHORITY_2**

Tyto vstupní body podporují použití identifikátoru zabezpečení Windows (NT SID NT).

Tyto názvy jsou definovány jako **typedef** v hlavičkovém souboru `cmqzc.h`, který lze použít k vytvoření prototypu funkcí komponent.

Inicializační funkce (**MQZ_INIT_AUTHORITY**) musí být hlavním vstupním bodem komponenty. Ostatní funkce jsou vyvolány přes adresu vstupního bodu, kterou inicializační funkce přidala do vektoru vstupního bodu komponenty.

Rozhraní služby názvů
Služba názvů poskytuje vstupní body pro použití správcem front.

K dispozici jsou následující vstupní body:

NÁZEV MQZ_INIT_NAME
Inicializovat komponentu služby názvů.

NÁZEV MQZ_TERM_NAME
Ukončete komponentu služby názvů.

NÁZEV MQZ_LOOKUP_NAME
Vyhledejte název správce front pro danou frontu.

MQZ_INSERT_NAME
Vložte položku obsahující název správce front, který je vlastníkem dané fronty, do adresáře používaného touto službou.

MQZ_DELETE_NÁZEV
Vymažte záznam pro uvedenou frontu z adresáře používaného službou.

Je-li nakonfigurována více než jedna služba názvů:

- Pro vyhledávání je vyvolána funkce MQZ_LOOKUP_NAME pro každou službu v seznamu, dokud nebude název fronty vyřešen (pokud některá komponenta neoznačuje, že by mělo hledání zastavit).
- Pro vložení je vyvolána funkce MQZ_INSERT_NAME pro první službu v seznamu, která podporuje tuto funkci.
- Pro odstranění je funkce MQZ_DELETE_NAME vyvolána pro první službu v seznamu, která podporuje tuto funkci.

Nemít více než jednu komponentu, která podporuje funkce vložení a odstranění. Avšak komponenta, která podporuje pouze vyhledávání, je proveditelná a lze ji použít například jako poslední komponentu v seznamu k vyřešení jakéhokoli názvu, který není znám žádnou jinou komponentou služby názvů, ke správci front, ve kterém lze definovat název.

V programovacím jazyku C jsou názvy definovány jako datové typy funkcí pomocí příkazu typedef. Lze je použít k vytvoření prototypu servisních funkcí, aby se zajistilo, že parametry jsou správné.

Soubor záhlaví, který obsahuje veškerý materiál specifický pro instalovatelné služby, je `cmqzc.h` pro jazyk C.

Kromě inicializační funkce (MQZ_INIT_NAME), která musí být hlavním vstupním bodem komponenty, jsou funkce vyvolány adresou vstupního bodu, kterou funkce inicializace přidala, pomocí volání MQZEP.

Použití více komponent služeb
Pro službu můžete instalovat více než jednu komponentu. To umožňuje komponentám poskytovat pouze částečné implementace služby a spolehnout se na ostatní komponenty, aby poskytly zbývající funkce.

Příklad použití více komponent

Předpokládejme, že vytvoříte dvě komponenty služby názvů nazvané `ABC_name_serv` a `XYZ_name_serv`.

`ABC_name_serv`
Tato komponenta podporuje vložení názvu do adresáře služeb nebo odstranění jeho názvu z adresáře služeb, ale nepodporuje vyhledávání názvu fronty.

`XYZ_name_serv`
Tato komponenta podporuje vyhledání názvu fronty, ale nepodporuje vložení názvu do adresáře služeb nebo odstranění názvu z adresáře služeb.

Komponenta `ABC_name_serv` uchovává databázi názvů front a používá dva jednoduché algoritmy k vložení nebo odstranění názvu z adresáře služby.

Komponenta `XYZ_name_serv` používá jednoduchý algoritmus, který vrací pevný název správce front pro všechny názvy front, s nimiž je vyvolána. Neobsahuje databázi názvů front, a proto nepodporuje funkce vkládání a odstraňování.

Komponenty jsou instalovány ve stejném správci front. Stanzy *`ServiceComponent`* jsou seřazeny tak, že komponenta `ABC_name_serv` je vyvolána jako první. Jakákoli volání pro vložení nebo odstranění fronty v adresáři komponenty jsou ošetřeny komponentou `ABC_name_serv` ; Je to jediná, která implementuje tyto funkce. Avšak volání vyhledání, které komponenta `ABC_name_serv` nemůže interpretovat, je předáno do komponenty pouze pro vyhledávání, `XYZ_name_serv`. Tato komponenta dodává název správce front z jednoduchého algoritmu.

Vynechání vstupních bodů při použití více komponent

Pokud se rozhodnete použít více komponent k poskytování služby, můžete navrhnout komponentu služby, která neimplementuje určité funkce. Rámec instalovatelných služeb neklade žádná omezení, na které můžete vynechat. Avšak v případě určitých instalovatelných služeb může být vynechání jedné nebo více funkcí logicky nekonzistentní s účelem služby.

Příklad vstupních bodů použitých s více komponentami

Tabulka 124 na stránce 908 ukazuje příklad instalovatelné služby názvů, pro kterou byly nainstalovány dvě komponenty. Každá podporuje jinou sadu funkcí přidružených k této konkrétní instalovatelné službě. Pro funkci vložení je nejprve vyvolán vstupní bod komponenty ABC. Vstupní body, které nebyly definovány pro službu (pomocí **MQZEP**) jsou považovány za NULL. Vstupní bod pro inicializaci je uveden v tabulce, ale to není povinné, protože inicializace je prováděna hlavním vstupním bodem komponenty.

Má-li správce front použít instalovatelnou službu, použije vstupní body definované pro tuto službu (sloupce v produktu Tabulka 124 na stránce 908). Při použití každé komponenty správce front určuje

adresu rutiny, která implementuje požadovanou funkci. To pak volá rutinu, pokud existuje. Je-li operace úspěšná, správce front použije všechny výsledky a informace o stavu.

Tabulka 124. Příklad vstupních bodů pro instalovatelnou službu

Číslo funkce	**Komponenta služby názvu ABC**	**Komponenta služby názvů XYZ**
MQZID_INIT_NAME (Inicializace)	Funkce ABC_initialize ()	XYZ_initialize ()
MQZID_TERM_NAME (Ukončení)	Funkce ABC_terminate ()	XYZ_terminate ()
MQZID_INSERT_NAME (Vložení)	Funkce ABC_Insert ()	NULL
MQZID_DELETE_NAME (Výmaz)	Funkce ABC_Delete ()	NULL
MQZID_LOOKUP_NAME (vyhledání)	NULL	XYZ_Vyhledávání ()

Pokud rutina neexistuje, bude tento proces ve správci front opakován pro další komponentu v seznamu. Kromě toho, pokud rutina existuje, ale vrací kód označující, že nemohl operaci provést, pokračuje pokus s další dostupnou komponentou. Rutiny v komponentách služeb mohou vrátit kód, který označuje, že by se neměly provádět žádné další pokusy o provedení operace.

Konfigurace služeb a komponent

Konfigurujte komponenty služeb pomocí konfiguračních souborů správce front, kromě systémů Windows , kde má každý správce front vlastní objekt stanza v registru.

1. Chcete-li definovat službu pro správce front a určit umístění modulu, přidejte do konfiguračního souboru správce front oddíly.

   Každá použitá služba musí mít objekt stanza *Service* , který definuje službu pro správce front.

   Pro každou komponentu v rámci služby musí existovat stanza *ServiceComponent* . Identifikuje název a cestu k modulu, který obsahuje kód dané komponenty.

   Další informace viz “Formát sekce služby” na stránce 908 a “Formát sekce komponent služby” na stránce 909

   Komponenta autorizační služba, známá jako OAM (Object Authority Manager), se dodává spolu s produktem. Při vytváření správce front je konfigurační soubor správce front (nebo registr v systému Windows) automaticky aktualizován tak, aby obsahoval příslušné oddíly pro autorizační službu a pro výchozí komponentu (OAM). Pro ostatní komponenty musíte nakonfigurovat konfigurační soubor správce front ručně.

   Kód pro každou komponentu služby je načten do správce front při spuštění správce front s použitím dynamické vazby, kde je tato podpora na platformě podporována.
2. Chcete-li aktivovat komponentu, zastavte a znovu spusťte správce front.

Formát sekce služby

Sekce Služba obsahuje název služby a počet vstupních bodů definovaných pro službu.

Formát stanzy je následující:

```
Service:
  Name=service_name
  EntryPoints=entries
  SecurityPolicy=policy
```

kde:

service_name
Název služby. Toto je definováno službou.

entries
Počet vstupních bodů definovaných pro službu. To zahrnuje vstupní body inicializace a ukončení.

policy

Linux UNIX Na systémech UNIX and Linux : `uživatel`, `skupina`nebo `výchozí`. Tato hodnota určuje, zda správce front používá autorizaci založenou na uživateli nebo na základě skupin. Hodnoty nejsou citlivé na velikost písmen. Pokud tento atribut nezahrnete, použije se výchozí hodnota autorizace založená na skupině. Aby se změny projevily, restartujte správce front. Další informace najdete v tématu “Konfigurace oddílů autorizační služby v systému UNIX and Linux” na stránce 909.

Windows V systémech Windows : `NTSIDsRequired` (Identifikátor zabezpečení produktu Windows) nebo `Výchozí`. Pokud neuvedete `NTSIDsRequired`, použije se hodnota `Default` . Tento atribut je platný pouze, pokud **Name** má hodnotu `AuthorizationService`. Další informace najdete v tématu “Konfigurace oddílů autorizační služby v systému Windows” na stránce 910.

Formát sekce komponent služby
Stanzy **Service** a **ServiceComponent** se mohou vyskytovat v libovolném pořadí.

Formát stanzy komponenty služby je:

```
ServiceComponent:
  Service=service_name
  Name=component_name
  Module=module_name
  ComponentDataSize=size
```

kde:

service_name
Název služby. Musí odpovídat `Name` uvedenému ve stanze služby.

component_name
Popisný název komponenty služby. Musí být jedinečný a obsahovat pouze znaky, které jsou platné pro názvy objektů produktu IBM MQ (například názvy front). Tento název se vyskytuje ve zprávách operátora generovaných službou. Doporučujeme použít jméno začínající ochrannou známkou společnosti nebo obdobným rozlišovacím řetězcem.

module_name
Název modulu, který má obsahovat kód pro tuto komponentu.

size
Velikost oblasti dat komponenty předané komponentě při každém volání v bajtech. Uveďte nulu, pokud nejsou požadována žádná data komponenty.

Stanzy **Service** a **ServiceComponent** se mohou vyskytnout v libovolném pořadí a klíče stanzy pod nimi se mohou také vyskytnout v libovolném pořadí. Pro každou z těchto stanz musí být všechny klíče oddílu přítomny. Je-li klíč oddílu duplikován, použije se poslední.

Při spuštění správce front zpracuje všechny položky komponenty služby v konfiguračním souboru postupně. Poté načte uvedený modul komponenty, vyvolá vstupní bod komponenty (která musí být vstupním bodem pro inicializaci komponenty) a předá ji obslužnou rutinu konfigurace.

Linux UNIX *Konfigurace oddílů autorizační služby v systému UNIX and Linux*
V systému UNIX and Linuxmá každý správce front vlastní konfigurační soubor správce front.

Například výchozí cesta a název souboru s konfiguračním souborem správce front pro správce front QMNAME je `/var/mqm/qmgrs/QMNAME/qm.ini`.

Stanza `Service` a stanza `ServiceComponent` pro výchozí autorizační komponentu se přidávají do `qm.ini` automaticky, ale mohou být přepsány `mqsnoaut`. Jakékoli další oddíly `ServiceComponent` musí být přidány ručně.

Například následující sekce v konfiguračním souboru správce front definují dvě komponenty autorizační služby v produktu IBM MQ for AIX. *MQ_INSTALLATION_PATH* představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

```
 Service:
    Name=AuthorizationService
    EntryPoints=13

 ServiceComponent:
    Service=AuthorizationService
    Name=MQSeries.UNIX.auth.service
Module= MQ_INSTALLATION_PATH/lib/amqzfu
    ComponentDataSize=0

 ServiceComponent:
    Service=AuthorizationService
    Name=user.defined.authorization.service
    Module=/usr/bin/udas01
    ComponentDataSize=96
```

Obrázek 110. Stanzy autorizační služby produktu UNIX and Linux v souboru qm.ini

Stanza service component (`MQSeries.UNIX.auth.service`) definuje výchozí komponentu autorizační služby, OAM. Pokud odeberete tuto stanzu a restartujete správce front, bude OAM zablokováno a nebudou provedeny žádné kontroly autorizace.

Windows *Konfigurace oddílů autorizační služby v systému Windows*

V systému IBM MQ for Windows má každý správce front svůj vlastní oddíl v registru.

Stanza *`Service`* a stanza *`ServiceComponent`* pro výchozí autorizační komponentu jsou přidány do registru automaticky, ale mohou být přepsány pomocí `mqsnoaut`. Jakékoli další oddíly *`ServiceComponent`* musí být přidány ručně.

Atribut `SecurityPolicy` můžete také přidat pomocí služeb produktu IBM MQ . Atribut `SecurityPolicy` se používá pouze v případě, že služba uvedená ve stanze *`Service`* je autorizační služba, to znamená výchozí OAM. Atribut `SecurityPolicy` vám umožňuje uvést zásady zabezpečení pro každého správce front. Možné hodnoty jsou:

`Default`
: Uveďte `Default` , pokud chcete, aby se projevila výchozí zásada zabezpečení. Pokud identifikátor zabezpečení produktu Windows (NT SID) není předán OAM pro konkrétní ID uživatele, provede se pokus o získání příslušného SID prohledáváním příslušných databází zabezpečení.

`NTSIDsRequired`
: Vyžaduje, aby při provádění kontrol zabezpečení byl při provádění kontroly zabezpečení předán identifikátor SID systému NT.

Informace o formátu sekce Service stanza viz “Formát sekce služby” na stránce 908. Další obecné informace o zabezpečení najdete v tématu Nastavení zabezpečení v systémech Windows, UNIX and Linux.

Stanza Service Component, `MQSeries.WindowsNT.auth.service` definuje výchozí komponentu autorizační služby, OAM. Pokud odeberete tuto stanzu a restartujete správce front, bude OAM zablokováno a nebudou provedeny žádné kontroly autorizace.

Linux UNIX *Konfigurace oddílů služby názvů: Systémy UNIX and Linux*

V systémech UNIX and Linux má každý správce front svůj vlastní konfigurační soubor správce front.

Následující příklady stanz konfiguračního souboru UNIX and Linux pro službu názvů určují komponentu služby názvů, kterou poskytla (fiktivní) společnost ABC.

```
# Stanza for name service
 Service:
    Name=NameService
    EntryPoints=5

# Stanza for name service component, provided by ABC
 ServiceComponent:
    Service=NameService
    Name=ABC.Name.Service
    Module=/usr/lib/abcname
    ComponentDataSize=1024
```

Obrázek 111. Stanzy služby názvů v souboru qm.ini (pro systémy UNIX and Linux)

Poznámka: Windows Na systémech Windows se informace o sekci služby názvů ukládají do registru.

Aktualizace OAM po změně autorizace uživatele
V produktu IBM MQmůžete aktualizovat informace o skupině autorizace OAM ihned po změně členství skupiny autorizace uživatele, což odráží změny provedené na úrovni operačního systému, aniž by bylo nutné zastavit a znovu spustit správce front. Chcete-li to provést, zadejte příkaz **REFRESH SECURITY** .

Poznámka: Když změníte autorizace pomocí příkazu `setmqaut` , produkt OAM tyto změny okamžitě naimplementuje.

Správci front ukládají autorizační data do lokální fronty s názvem SYSTEM.AUTH.DATA.QUEUE. Tato data jsou spravována produktem **amqzfuma.exe**.

Související informace
REFRESH SECURITY

IBM i Instalovatelné služby a komponenty v systému IBM i

Tyto informace použijte k seznámení se s instalovatelnými službami a s funkcemi a komponentami, které jsou k nim přidruženy. Rozhraní pro tyto funkce je dokumentováno tak, že vy nebo dodavatelé softwaru můžete dodávat komponenty.

Hlavní důvody pro poskytování instalovatelných služeb produktu IBM MQ jsou:

- Chcete-li vám poskytnout flexibilitu při výběru toho, zda mají být použity komponenty poskytované produktem IBM MQ for IBM i, nebo je můžete nahradit nebo rozšířit o jiné komponenty.
- Chcete-li dodavatelům umožnit účast, tím, že poskytnete komponenty, které mohou využívat nové technologie, aniž byste provedli interní změny v produktu IBM MQ for IBM i.
- Umožněte produktu IBM MQ využívat rychlejší a levnější využití nových technologií, a proto poskytovat produkty dříve a za nižší ceny.

Instalovatelné služby a *komponenty služeb* jsou součástí struktury produktu IBM MQ . Ve středu této struktury je ta část správce front, která implementuje danou funkci a pravidla přidružená k rozhraní MQI (Message Queue Interface). Tato centrální část vyžaduje řadu servisních funkcí nazývaných *instalovatelné služby*, aby mohla provést svou práci. Instalovatelná služba, která je k dispozici v produktu IBM MQ for IBM i , je autorizační služba.

Každá instalovatelná služba je související sadou funkcí implementovaných pomocí jedné nebo více *komponent služeb*. Každá komponenta je vyvolána pomocí veřejně dostupného rozhraní, které je k dispozici. To umožňuje nezávislým dodavatelům softwaru a jiným třetím stranám poskytovat instalovatelné komponenty k rozšíření nebo nahrazení těch, které poskytuje produkt IBM MQ for IBM i. Tabulka 125 na stránce 912 shrnuje podporu pro ověřovací službu.

Tabulka 125. Souhrn komponent ověřovací služby		
Dodaná komponenta	**Funkce**	**Požadavky**
správce oprávnění k objektu (OAM)	Poskytuje kontrolu autorizace pro příkazy a volání MQI. Uživatelé mohou napsat svou vlastní komponentu pro rozšíření nebo nahrazení OAM.	(Předpokládá se vhodná oprávnění k autorizaci platformy)
Komponenta služby názvů DCE **Poznámka:** DCE je podporováno pouze ve verzích produktu IBM MQ starších než 6.0.	• Umožňuje správcům front sdílet fronty nebo • Definované uživatelem **Poznámka:** Sdílené fronty musí mít nastaven atribut **Scope** nastaven na hodnotu CELL.	• Je vyžadováno prostředí DCE pro dodanou komponentu, nebo • třetí strana nebo uživatel zapsaný jménem uživatele

IBM i *Funkce a komponenty v systému IBM i*

Tyto informace použijte k pochopení funkcí a komponent, vstupních bodů, návratových kódů a dat komponent, které můžete použít v produktu IBM MQ for IBM i.

Každá služba se skládá ze sady souvisejících funkcí. Např. služba názvů obsahuje funkci pro:

- Probíhá hledání názvu fronty a vrácení názvu správce front, ve kterém je fronta definována.
- Vložení názvu fronty do adresáře služby
- Odstranění názvu fronty z adresáře služby

Obsahuje také funkce inicializace a ukončení.

Instalovatelná služba je poskytována jednou nebo více komponentami služeb. Každá komponenta může provádět některé nebo všechny funkce, které jsou pro danou službu definovány. Komponenta je také zodpovědná za správu veškerých podkladových prostředků nebo softwaru, které potřebuje pro implementaci služby. Konfigurační soubory poskytují standardní způsob, jak načíst komponentu a určit adresy funkčních rutin, které poskytuje.

Služby a komponenty se vztahují takto:

- Služba je definována pro správce front podle oddílů v konfiguračním souboru.
- Každá služba je podporována zadaným kódem ve správci front. Uživatelé nemohou tento kód změnit, a proto nemohou vytvořit své vlastní služby.
- Každá služba je implementována jednou nebo více komponentami. Tyto služby mohou být dodány spolu s produktem nebo uživatelem napsanými. Je možné vyvolat více komponent pro službu, přičemž každá z nich podporuje různá zařízení v rámci služby.
- Vstupní body spojují komponenty služeb s podpůrným kódem ve správci front.

Vstupní body

Každá komponenta služby je představována pomocí seznamu adres vstupního bodu rutin, které podporují konkrétní instalovatelnou službu. Instalovatelná služba definuje funkci, která má být prováděna každou rutinou. Uspořádání komponent služeb, když jsou nakonfigurovány, definuje pořadí, ve kterém jsou vstupní body volány ve snaze vyhovět požadavku na službu. V dodaném hlavičkovém souboru `cmqzc.h` mají dodávané vstupní body pro každou službu předponu MQZID_.

Návratové kódy

Komponenty služeb poskytují správci front návratové kódy pro vytváření sestav o různých podmínkách. Ohlašují úspěch nebo selhání operace a označují, zda má správce front pokračovat do další komponenty služby. Samostatná hodnota parametru *Continuation* je tato indikace.

Data komponent

Jedna komponenta služby může vyžadovat sdílení dat mezi různými funkcemi. Instalovatelné služby poskytují volitelnou datovou oblast, která má být předána při každém vyvolání určité komponenty služby. Tato datová oblast je určena pro výlučné použití komponenty služby. Je sdílen všemi vyvoláními dané funkce i v případě, že jsou vytvořeny z různých adresních prostorů nebo procesů. Je zaručeno, že bude adresovatelný z komponenty služby, kdykoli se zavolá. Je třeba deklarovat velikost této oblasti ve stanze *ServiceComponent* .

IBM i *Inicializace v systému IBM i*

Při vyvolání inicializační rutiny komponenty musí být volána funkce MQZEP správce front pro každý vstupní bod podporovaný danou komponentou. MQZEP definuje vstupní bod ke službě. Předpokládá se, že všechny nedefinované výstupní body jsou NULL.

Primární inicializace
Komponenta je vždy vyvolána s touto volbou jednou, před tím, než je vyvolána jiným způsobem.

Sekundární inicializace
S touto volbou lze na určitých platformách vyvolat určitou komponentu. Může být například vyvolán jednou pro každý proces operačního systému, podproces nebo úlohu, ke které je služba přistupována.

Je-li použita sekundární inicializace:

- Komponenta může být vyvolána více než jednou pro sekundární inicializaci. Pro každé takové volání se vydá odpovídající volání pro sekundární ukončení, když již služba není potřebná.

  U autorizačních služeb se jedná o volání MQZ_TERM_AUTHORITY.
- Vstupní body musí být znovu zadány (voláním MQZEP) pokaždé, když je komponenta volána pro primární a sekundární inicializaci.
- Pro komponentu se použije pouze jedna kopie dat komponenty; pro každou sekundární inicializaci není jiná kopie.
- Komponenta není vyvolána pro žádné další volání na službu (z procesu operačního systému, vlákna nebo úlohy, jak je to vhodné) před sekundární inicializací.
- Komponenta musí nastavit parametr **Version** na stejnou hodnotu pro primární a sekundární inicializaci.

Primární ukončení
Komponenta je vždy spuštěna s touto volbou jednou, když již není potřeba. K této komponentě nejsou vytvořena žádná další volání.

Sekundární ukončení
Komponenta je spuštěna s touto volbou, pokud byla spuštěna pro sekundární inicializaci.

IBM i *Konfigurace služeb a komponent v systému IBM i*

Konfigurujte komponenty služeb pomocí konfiguračních souborů správce front. Každá použitá služba musí mít objekt stanza *Service* , který definuje službu pro správce front.

Pro každou komponentu v rámci služby musí existovat stanza *ServiceComponent* . Identifikuje název a cestu k modulu, který obsahuje kód dané komponenty.

Komponenta autorizační služba, známá jako OAM (Object Authority Manager), se dodává spolu s produktem. Při vytváření správce front je konfigurační soubor správce front automaticky aktualizován tak, aby obsahoval příslušné oddíly pro autorizační službu a pro výchozí komponentu (OAM).

Kód pro každou komponentu služby je načten do správce front při spuštění správce front s použitím dynamické vazby, kde je tato podpora na platformě podporována.

Formát sekce služby

Formát stanzy **Service** je:

```
Service:
  Name=service_name
  EntryPoints=entries
```

kde:

service_name
Název služby. Toto je definováno službou.

entries
Počet vstupních bodů definovaných pro službu. To zahrnuje vstupní body inicializace a ukončení.

Formát sekce komponent služby

Formát stanzy **Service component** je:

```
ServiceComponent:
  Service=service_name
  Name=component_name
  Module=module_name
  ComponentDataSize=size
```

kde:

service_name
Název služby. Musí odpovídat atributu *Name* uvedenému ve stanze služby.

component_name
Popisný název komponenty služby. Musí být jedinečný a obsahovat pouze znaky, které jsou platné pro názvy objektů produktu IBM MQ (například názvy front). Tento název se vyskytuje ve zprávách operátora generovaných službou. Doporučujeme použít jméno začínající ochrannou známkou společnosti nebo obdobným rozlišovacím řetězcem.

module_name
Název modulu, který má obsahovat kód pro tuto komponentu. Uveďte úplný název cesty.

size
Velikost oblasti dat komponenty předané komponentě při každém volání v bajtech. Uveďte nulu, pokud nejsou požadována žádná data komponenty.

Tyto dvě stanzy se mohou vyskytnout v libovolném pořadí a klíče stanzy pod nimi se mohou objevit také v libovolném pořadí. Pro každou z těchto stanz musí být všechny klíče oddílu přítomny. Je-li klíč oddílu duplikován, použije se poslední.

Při spuštění správce front zpracuje všechny položky komponenty služby v konfiguračním souboru postupně. Poté načte uvedený modul komponenty, vyvolá vstupní bod komponenty (která musí být vstupním bodem pro inicializaci komponenty) a předá ji obslužnou rutinu konfigurace.

IBM i *Vytvoření vlastní komponenty služby v systému IBM i*

Pomocí těchto informací se naučíte, jak vytvořit komponentu služby v produktu IBM MQ for IBM i.

Chcete-li vytvořit vlastní komponentu služby:

- Ujistěte se, že hlavičkový soubor `cmqzc.h` je zahrnut ve vašem programu.
- Vytvořte sdílenou knihovnu kompilací programu a propojte ji se sdílenými knihovnami `libmqm*` a `libmqmzf*`.

  Poznámka: Protože se agent může spustit v prostředí s podprocesy, musíte sestavit OAM, aby se spouštěl v prostředí s podprocesy. To zahrnuje použití podprocesů se závitem `libmqm` a `libmqmzf`.
- Chcete-li definovat službu pro správce front a určit umístění modulu, přidejte oddíly do konfiguračního souboru správce front.

- Chcete-li aktivovat komponentu, zastavte a znovu spusťte správce front.

IBM i *Autorizační služba na systému IBM i*

Autorizační služba je instalovatelná služba, která umožňuje správcům front vyvolávat autorizační prostředky, například kontrolu, zda má ID uživatele oprávnění k otevření fronty.

Tato služba je komponenta zabezpečeného rozhraní produktu IBM MQ (SEI), která je součástí rámce produktu IBM MQ . Prodiskutují se následující témata:

- “správce oprávnění k objektu (OAM)” na stránce 915
- “Definování služby pro operační systém” na stránce 915
- “Konfigurace oddílů autorizační služby” na stránce 915
- “Rozhraní autorizační služby v systému IBM i” na stránce 916

správce oprávnění k objektu (OAM)

Komponenta autorizační služba dodaná s produkty IBM MQ se nazývá správce oprávnění k objektu (OAM). Ve výchozím nastavení je OAM aktivní a pracuje s následujícími řídicími příkazy:

- **WRKMQMAUT** práce s oprávněním
- **WRKMQMAUTD** práce s daty oprávnění
- Oprávnění k zobrazení objektu **DSPMQMAUT**
- **GRTMQMAUT** udělit oprávnění k objektu
- **RVKMQMAUT** odvolat oprávnění k objektu
- **RFRMQMAUT** REFRESH SECURITY

Syntaxe těchto příkazů a informace o jejich použití jsou popsány v nápovědě k příkazu CL. OAM pracuje s *entitou* činitele nebo skupiny.

Je-li vydán požadavek na rozhraní MQI nebo je vydán příkaz, zkontroluje produkt OAM autorizaci entity přidružené k operaci a zjistí, zda může provést následující akce:

- Proveďte požadovanou operaci.
- Přistupte k určeným prostředkům správce front.

Autorizační služba vám umožňuje rozšířit nebo nahradit kontrolu oprávnění poskytovanou pro správce front tím, že zapisujete svou vlastní komponentu autorizační služby.

Definování služby pro operační systém

Stanzy autorizační služby v konfiguračním souboru správce front `qm.ini` definují autorizační službu pro správce front. Chcete-li získat informace o typech sekcí, prohlédněte si příručku “Konfigurace služeb a komponent v systému IBM i” na stránce 913 .

Konfigurace oddílů autorizační služby

V systému IBM MQ for IBM i:

Hlavní
Je profil uživatele systému IBM i .

Skupina
Je profil skupiny systému IBM i .

Oprávnění lze udělovat nebo odvolat pouze na úrovni skupiny. Požadavek na udělení nebo odebrání oprávnění uživatele aktualizuje primární skupinu pro tohoto uživatele.

Každý správce front má svůj vlastní konfigurační soubor správce front. Například výchozí cesta a název souboru s konfiguračním souborem správce front pro správce front QMNAME je `/QIBM/UserData/mqm/qmgrs/QMNAME/qm.ini`.

Stanza *Service* a stanza *ServiceComponent* pro výchozí autorizační komponentu se přidávají do `qm.ini` automaticky, ale mohou být přepsány WRKENVVAR. Jakékoli další oddíly *ServiceComponent* musí být přidány ručně.

Například následující sekce v konfiguračním souboru správce front definují dvě komponenty autorizační služby:

```
Service:
    Name=AuthorizationService
    EntryPoints=7

ServiceComponent:
    Service=AuthorizationService
    Name=MQ.UNIX.authorization.service
    Module=QMQM/AMQZFU
    ComponentDataSize=0

ServiceComponent:
    Service=AuthorizationService
    Name=user.defined.authorization.service
    Module=LIBRARY/SERVICE PROGRAM NAME
    ComponentDataSize=96
```

Obrázek 112. Stanzy autorizační služby v souboru qm.ini v systému IBM i

Oddíl první služby komponenty `MQ.UNIX.authorization.service` definuje výchozí komponentu autorizační služby, OAM. Pokud odeberete tuto stanzu a restartujete správce front, bude OAM zablokováno a nebudou provedeny žádné kontroly autorizace.

IBM i *Rozhraní autorizační služby v systému IBM i*

Rozhraní autorizační služby poskytuje několik vstupních bodů pro použití správcem front.

MQZ_AUTHENTICATE_USER
Ověřuje ID uživatele a heslo a může nastavit pole kontextu identity.

MQZ_CHECK_AUTHORITY
Kontroluje, zda má entita oprávnění provést jednu nebo více operací na uvedeném objektu.

MQZ_COPY_ALL_AUTHORITY
Kopíruje všechna aktuální oprávnění, která existují pro odkazovaný objekt, na jiný objekt.

OPRÁVNĚNÍ MQZ_DELETE_AUTHORITY
Odstraní všechny autorizace přidružené k uvedenému objektu.

MQZ_ENUMERATE_AUTHORITY_DATA
Načte všechna data oprávnění, která se shodují s uvedenými kritérii výběru.

MQZ_FREE_USER
Uvolní přidružené přidělené prostředky.

FUNKCE MQZ_GET_AUTHORITY
Získá oprávnění, které má entita pro přístup k uvedenému objektu.

MQZ_GET_EXPLICITNÍ_AUTORITA
Získá buď oprávnění, které má pojmenovaná skupina k přístupu k uvedenému objektu (ale bez dalšího oprávnění skupiny **nikdo**), nebo oprávnění, které má primární skupina uvedeného hlavního objektu k přístupu k uvedenému objektu.

MQZ_INIT_AUTHORITY
Inicializuje komponentu autorizační služby.

MQZ_DOTÁZAT SE
Dotáže se podporované funkčnosti autorizační služby.

MQZ_REFRESH_CACHE
Aktualizujte všechny autorizace.

OPRÁVNĚNÍ MQZ_SET_AUTHORITY
Nastaví oprávnění, které má entita k uvedenému objektu.

OPRÁVNĚNÍ MQZ_TERM_AUTHORITY
Ukončí komponentu autorizační služby.

Tyto vstupní body podporují použití identifikátoru zabezpečení Windows (NT SID NT).

Tyto názvy jsou definovány jako **typedef** v hlavičkovém souboru `cmqzc.h`, který lze použít k vytvoření prototypu funkcí komponent.

Inicializační funkce (**MQZ_INIT_AUTHORITY**) musí být hlavním vstupním bodem komponenty. Ostatní funkce jsou vyvolány přes adresu vstupního bodu, kterou inicializační funkce přidala do vektoru vstupního bodu komponenty.

Další informace viz "Vytvoření vlastní komponenty služby v systému IBM i" na stránce 914.

Zápis a kompilace uživatelských procedur rozhraní API

Uživatelské procedury rozhraní API vám umožňují psát kód, který změní chování volání rozhraní API produktu IBM MQ , jako je například MQPUT a MQGET, a pak tento kód vloží bezprostředně před nebo bezprostředně za těmito voláními.

Poznámka: Nepodporováno na IBM MQ for z/OS.

Proč používat uživatelské procedury rozhraní API?

Každá z vašich aplikací má specifickou úlohu a její kód by měl provést tuto úlohu co nejefektivněji. Na vyšší úrovni byste mohli chtít použít standardy nebo obchodní procesy pro konkrétního správce front pro **všechny** aplikace, které tento správce front používají. Je účinnější provádět tuto úroveň nad úrovní jednotlivých aplikací, a tudíž bez nutnosti měnit kód každé ovlivněné aplikace.

Zde je několik návrhů oblastí, ve kterých mohou být uživatelské procedury rozhraní API užitečné:

- V případě *zabezpečení*můžete poskytnout ověření a zkontrolovat, zda jsou aplikace autorizovány pro přístup ke správci front nebo ke správci front. Můžete také použít rozhraní API pro práci s policejními aplikacemi, ověřovat jednotlivá volání rozhraní API nebo dokonce i parametry, které používají.
- Pro *flexibilitu*můžete reagovat na rychlé změny ve vašem obchodním prostředí, aniž byste změnili aplikace, které spoléhají na data v daném prostředí. Mohli byste například mít uživatelské procedury rozhraní API, které reagují na změny úrokových sazeb, směnných kurzů měn nebo ceny komponent ve výrobním prostředí.
- Pro *monitorování* použití fronty nebo správce front můžete trasovat tok aplikací a zpráv, chyby v protokolu v voláních rozhraní API, nastavit záznamy monitorování pro účely evidence nebo shromažďovat statistiky využití pro účely plánování.

Co se stane, když se spustí uživatelská procedura rozhraní API?

Jakmile jste napsali výstupní program a identifikovali jej do produktu IBM MQ, správce front automaticky vyvolá váš výstupní kód v registrovaných bodech.

Rutiny ukončení rozhraní API, které se mají spustit, jsou identifikovány ve stanzách v systémech IBM i , Windows, UNIX and Linux . Toto téma se vztahuje na sekce v konfiguračních souborech mqs.ini a qm.ini.

Definice rutin se může vyskytnout na třech místech:

1. ApiExitCommon, v souboru mqs.ini , identifikuje rutiny, pro celé IBM MQ, aplikované při spuštění správců front. Ty mohou být přepsány rutinami definovanými pro jednotlivé správce front (viz položka"3" na stránce 917v tomto seznamu).
2. Šablona ApiExitv souboru mqs.ini identifikuje rutiny pro celý soubor IBM MQ, které jsou zkopírovány do lokální sady ApiExit(viz položka "3" na stránce 917 v tomto seznamu) při vytvoření nového správce front.
3. ApiExitLokální, v souboru qm.ini , identifikuje rutiny, které se vztahují na konkrétního správce front.

Při vytvoření nového správce front se definice šablon ApiExitv souboru mqs.ini zkopírují do lokálních definic ApiExitv souboru qm.ini nového správce front. Když je spuštěn správce front, jsou použity

jak lokální definice ApiExit, tak i lokální definice ApiExit. Lokální definice ApiExitnahrazují obecné definice ApiExit, pokud obě identifikují rutinu se stejným názvem. Atribut Sequence , který je popsán v “Konfigurace uživatelských procedur rozhraní API” na stránce 923 , určuje pořadí, ve kterém jsou rutiny definované ve stanzách spuštěny.

Použití uživatelských procedur rozhraní API ve více instalacích produktu IBM MQ

Ujistěte se, že uživatelské procedury rozhraní API zapsané pro dřívější verzi IBM MQ se používají pro práci se všemi verzemi, protože změny, které byly provedeny pro ukončení v produktu IBM WebSphere MQ 7.1 , nemusí fungovat s dřívější verzí. Další informace o změnách provedených pro ukončení naleznete v tématu “Zapisování uživatelských procedur a instalovatelných služeb na UNIX, Linux a Windows” na stránce 898.

Ukázky poskytnuté pro uživatelské procedury rozhraní API amqsaem a amqsaxe odrážejí změny vyžadované při zápisu uživatelských procedur. Aplikace klienta musí zajistit, aby před spuštěním aplikace byly propojeny správné knihovny produktu IBM MQ , které odpovídají instalaci správce front, k němuž je aplikace přidružena,.

Zápis uživatelských procedur API

Uživatelské procedury můžete zapsat pro každé volání API pomocí programovacího jazyka C.

Uživatelské procedury jsou k dispozici pro každé volání rozhraní API takto:

- MQCB, chcete-li znovu registrovat zpětné volání pro zadaný popisovač objektu a řídicí aktivaci a změny pro zpětné volání
- MQCTL, k provedení řídicích akcí na manipulátorech objektů otevřených pro připojení
- MQCONN/MQCONNX, který poskytuje manipulátor připojení správce front pro použití při následných voláních rozhraní API
- MQDISC, k odpojení od správce front
- MQBEGIN, chcete-li zahájit globální pracovní jednotku (UOW)
- MQBACK, chcete-li zálohovat jednotku UOW
- MQCMIT, k potvrzení jednotky UOW
- MQOPEN, chcete-li otevřít prostředek IBM MQ pro následný přístup
- MQCLOSE, chcete-li zavřít prostředek IBM MQ , který byl dříve otevřen pro přístup
- MQGET, chcete-li načíst zprávu z fronty, která byla dříve otevřena pro přístup
- MQPUT1, chcete-li umístit zprávu do fronty
- MQPUT, chcete-li umístit zprávu do fronty, která byla dříve otevřena pro přístup
- MQINQ, chcete-li se dotázat na atributy prostředku IBM MQ , který byl dříve otevřen pro přístup
- MQSET, chcete-li nastavit atributy fronty, která byla dříve otevřena pro přístup
- MQSTAT, chcete-li načíst informace o stavu
- MQSUB, chcete-li registrovat odběr aplikací pro konkrétní téma
- MQSUBRQ, chcete-li provést požadavek na odběr

MQ_CALLBACK_EXIT poskytuje funkci ukončení, která se má provést před a po zpracování zpětného volání. Další informace viz Callback-MQ_CALLBACK_EXIT.

V rámci uživatelských procedur rozhraní API má volání obecnou podobu:

```
MQ_call_EXIT (parameters, context, ApiCallParameters)
```

kde *call* je název volání MQI bez předpony MQ ; např. PUT, GET. Ovládací prvek *parameters* řídí funkci uživatelské procedury, která primárně poskytuje komunikaci mezi ukončením a externím řídicím blokům MQAXP (struktura výstupních parametrů rozhraní API) a MQAXC (struktura kontextu uživatelské

procedury rozhraní API). *context* popisuje kontext, ve kterém byla volána uživatelská procedura rozhraní API, a *ApiCallParameters* představují parametry pro volání MQI.

Při zápisu uživatelské procedury rozhraní API je k dispozici ukázkový výstup amqsaxe0.c; tento výstup generuje trasovací záznamy do souboru, který jste zadali. Tuto ukázku můžete použít jako výchozí bod při zápisu uživatelských procedur. Další informace o použití ukázkové uživatelské procedury naleznete v příručce "Ukázkový program uživatelské procedury rozhraní API" na stránce 1051.

Další informace o voláních uživatelské procedury rozhraní API, externích řídicích blocích a přidružených tématech naleznete v tématu Odkaz na ukončení rozhraní API.

Obecné informace o tom, jak zapisovat, kompilovat a konfigurovat ukončení, naleznete v části "Zapisování uživatelských procedur a instalovatelných služeb na UNIX, Linux a Windows" na stránce 898.

Používání popisovačů zpráv v uživatelských procedurách rozhraní API

Můžete řídit, ke kterým vlastnostem zprávy má přístup rozhraní API přístup. Vlastnosti jsou přidruženy k manipulátoru ExitMsg. Vlastnosti nastavené ve výstupní frontě jsou nastaveny na vkládanou zprávu, ale vlastnosti načtené v rámci procedury get exit se do aplikace nevrátí.

Pokud registrujete uživatelskou funkci MQ_INIT_EXIT pomocí volání MQXEP MQI s parametrem **Function** nastaveným na hodnotu MQXF_INIT a **ExitReason** nastaveným na hodnotu MQXR_CONNECTION, předáváte strukturu MQXEPO jako parametr **ExitOpts** . Struktura MQXEPO obsahuje pole ExitProperties , které určuje sadu vlastností, které mají být zpřístupněny pro ukončení. Je zadán jako znakový řetězec reprezentující předponu vlastností, která odpovídá názvu složky MQRFH2 .

Každá uživatelská procedura rozhraní API přijímá strukturu MQAXP obsahující pole manipulátoru ExitMsg. Toto pole je nastaveno na hodnotu vygenerovanou serverem IBM MQ a je specifická pro připojení. Popisovač je tedy nezměněn mezi uživatelskými procedurami rozhraní API stejného nebo různých typů ve stejném připojení.

V příkazu MQ_PUT_EXIT nebo MQ_PUT1_EXIT s **ExitReason** MQXR_BEFORE, to znamená ukončení rozhraní API před vložením zprávy, jakékoli vlastnosti (jiné než vlastnosti deskriptoru zpráv) přidružené k obslužné rutiny ExitMsg, když je dokončení uživatelské procedury nastaveno na vkládané zprávy. Chcete-li tomu zabránit, nastavte ovladač ExitMsgna hodnotu MQHM_NONE. Můžete také dodat jiný popisovač zprávy.

V proměnné MQ_GET_EXIT a MQ_CALLBACK_EXIT je popisovač ExitMsgvymazán z vlastností a naplněn vlastnostmi uvedenými v poli ExitProperties , když byla registrována hodnota MQ_INIT_EXIT, jiná než vlastnosti deskriptoru zpráv. Tyto vlastnosti nejsou k dispozici pro získání aplikace. Je-li aplikace pro získání zprávy v poli MQGMO (Získat volby zprávy) zadána, jsou pro uživatelskou proceduru rozhraní API k dispozici všechny vlastnosti přidružené k tomuto popisovači včetně vlastností deskriptoru zpráv. Chcete-li zabránit tomu, aby byl popisovač ExitMsgnaplněn vlastnostmi, nastavte jej na hodnotu MQHM_NONE.

Poznámka: Pro vlastnosti ukončení zpráv, které mají být zpracovány v:

- Po funkci MQ_GET_EXIT je třeba definovat před ukončením funkce MQ_GET_EXIT pro ukončení.
- Před funkcí MQ_CALLBACK_EXIT je třeba definovat před ukončením funkce MQ_CB_EXIT pro ukončení.

K dispozici je ukázkový program amqsaem0.c, který ilustruje použití obslužných rutin zpráv v uživatelských procedurách rozhraní API.

Kompilace uživatelských procedur rozhraní API

Po zápisu uživatelské procedury zkompilujete a propojíte jej následujícím způsobem.

Následující příklady zobrazují příkazy použité pro ukázkový program popsaný v části "Ukázkový program uživatelské procedury rozhraní API" na stránce 1051. Pro jiné platformy než systémy Windows můžete najít ukázkový kód ukončení rozhraní API v produktu *MQ_INSTALLATION_PATH*/samp a v *MQ_INSTALLATION_PATH*/samp/binzkompilovanou a propojenou sdílenou knihovnu. Pro systémy Windows můžete nalézt vzorový kód ukončení rozhraní API v produktu *MQ_INSTALLATION_PATH* \Tools\c\Samples. *MQ_INSTALLATION_PATH* představuje adresář, ve kterém byl nainstalován produkt IBM MQ .

Poznámka pro uživatele:

1. Pokyny pro programování 64bitových aplikací jsou uvedeny v tématu Kódování standardů na 64bitových platformách

V případě zavedení klientů výběrového vysílání je možné na straně klienta spustit uživatelské procedury rozhraní API a uživatelské procedury pro převod dat, protože některé zprávy nemusí procházet správcem front. Následující knihovny jsou nyní součástí balíků klienta stejně jako balíky serveru:

Tabulka 126. Knihovny, které jsou nyní v balících klienta a serveru	
Operační systém	**Knihovny**
Windows	32 bit & 64 bit: mqm.dll & mqm.pdb
Linux & HP-UX	32 bit & 64 bit: libmqm.so & libmqm_r.so
AIX	32 bit & 64 bit: libmqm.a & libmqm_r.a
Solaris	32 bitů & 64 bitů: libmqm.so
IBM i	LIBMQM & LIBMQM_R

Kompilace uživatelských procedur rozhraní API na systémech Unix a Linux
Příklady způsobu kompilace uživatelských procedur rozhraní API v systémech UNIX and Linux .

Na všech platformách je vstupní bod do modulu MQStart.

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

zapAIX

Zkompilujte zdrojový kód uživatelské procedury rozhraní API zadáním jednoho z následujících příkazů:

32bitové aplikace

Nevláknová

```
cc -e MQStart -bE:amqsaxe.exp -bM:SRE -o /var/mqm/exits/amqsaxe \
amqsaxe0.c -I MQ_INSTALLATION_PATH/inc
```

Vláknové

```
xlc_r -e MQStart -bE:amqsaxe.exp -bM:SRE -o /var/mqm/exits/amqsaxe_r \
amqsaxe0.c -I MQ_INSTALLATION_PATH/inc
```

64bitové aplikace

Nevláknová

```
cc -q64 -e MQStart -bE:amqsaxe.exp -bM:SRE -o /var/mqm/exits64/amqsaxe \
amqsaxe0.c -I MQ_INSTALLATION_PATH/inc
```

Vláknové

```
xlc_r -q64 -e MQStart -bE:amqsaxe.exp -bM:SRE -o /var/mqm/exits64/amqsaxe_r \
amqsaxe0.c -I MQ_INSTALLATION_PATH/inc
```

Na platformě HP-UX Itanium

32bitové aplikace

Nevláknová

Kompilace zdrojového kódu ukončení API:

```
c89 +e +z -c -D_HPUX_SOURCE -o amqsaxe.o amqsaxe0.c -I MQ_INSTALLATION_PATH/inc
```

Odkaz na zdrojový kód uživatelské procedury rozhraní API

```
ld +b: -b amqsaxe.o +ee MQStart -o /var/mqm/exits/amqsaxe
rm amqsaxe.o
```

Vláknové

Kompilace zdrojového kódu ukončení API:

```
c89 +e +z -c -D_HPUX_SOURCE -o amqsaxe.o amqsaxe0.c -I MQ_INSTALLATION_PATH/inc
```

Odkaz na zdrojový kód uživatelské procedury rozhraní API

```
ld +b: -b amqsaxe.o +ee MQStart -o /var/mqm/exits/amqsaxe_r
rm amqsaxe.o
```

64bitové aplikace

Nevláknová

Kompilace zdrojového kódu ukončení API:

```
c89 +DD64 +e +z -c -D_HPUX_SOURCE -o amqsaxe.o amqsaxe0.c -I MQ_INSTALLATION_PATH/inc
```

Odkaz na zdrojový kód uživatelské procedury rozhraní API

```
ld -b amqsaxe.o +ee MQStart -o /var/mqm/exits64/amqsaxe
rm amqsaxe.o
```

Vláknové

Kompilace zdrojového kódu ukončení API:

```
c89 +DD64 +e +z -c -D_HPUX_SOURCE -o amqsaxe.o amqsaxe0.c -I MQ_INSTALLATION_PATH/inc
```

Odkaz na zdrojový kód uživatelské procedury rozhraní API

```
ld -b amqsaxe.o +ee MQStart -o /var/mqm/exits64/amqsaxe_r
rm amqsaxe.o
```

zapLinux

Zkompilujte zdrojový kód uživatelské procedury rozhraní API zadáním jednoho z následujících příkazů:

31bitové aplikace

Nevláknová

```
gcc -m31 -shared -fPIC -o /var/mqm/exits/amqsaxe amqsaxe0.c \
    -I MQ_INSTALLATION_PATH/inc
```

Vláknové

```
gcc -m31 -shared -fPIC -o /var/mqm/exits/amqsaxe_r amqsaxe0.c \
    -I MQ_INSTALLATION_PATH/inc
```

32bitové aplikace

Nevláknová

```
gcc -m32 -shared -fPIC -o /var/mqm/exits/amqsaxe amqsaxe0.c \
    -I MQ_INSTALLATION_PATH/inc
```

Vláknové

```
gcc -m32 -shared -fPIC -o /var/mqm/exits/amqsaxe_r amqsaxe0.c \
    -I MQ_INSTALLATION_PATH/inc
```

64bitové aplikace

Nevláknová

```
gcc -m64 -shared -fPIC -o /var/mqm/exits64/amqsaxe amqsaxe0.c \
    -I MQ_INSTALLATION_PATH/inc
```

Vláknové

```
gcc -m64 -shared -fPIC -o /var/mqm/exits64/amqsaxe_r amqsaxe0.c \
    -I MQ_INSTALLATION_PATH/inc
```

zapSolaris

Zkompilujte zdrojový kód uživatelské procedury rozhraní API zadáním jednoho z následujících příkazů:

32bitové aplikace

Platforma SPARC

```
cc -xarch=v8plus -KPIC -mt -G -o /var/mqm/exits/amqsaxe \
   amqsaxe0.c -I MQ_INSTALLATION_PATH/inc \
   -R/usr/lib/32 -lsocket -lnsl -ldl
```

Platformax86-64

```
cc -xarch=386 -KPIC -mt -G -o /var/mqm/exits/amqsaxe \
   amqsaxe0.c -I MQ_INSTALLATION_PATH/inc \
   -R/usr/lib/32 -lsocket -lnsl -ldl
```

64bitové aplikace

Platforma SPARC

```
cc -xarch=v9 -KPIC -mt -G -o /var/mqm/exits64/amqsaxe \
   amqsaxe0.c -I MQ_INSTALLATION_PATH/inc \
   -R/usr/lib/64 -lsocket -lnsl -ldl
```

Platformax86-64

```
cc -xarch=amd64 -KPIC -mt -G -o /var/mqm/exits64/amqsaxe \
   amqsaxe0.c -I MQ_INSTALLATION_PATH/inc \
   -R/usr/lib/64 -lsocket -lnsl -ldl
```

Na systémech Windows

Kompilujte a propojte ukázkový výstupní program rozhraní API, `amqsaxe0.c`, na Windows

Soubor typu manifest je volitelný dokument XML obsahující verzi nebo jinou informaci, která může být vložena do kompilované aplikace nebo knihovny DLL.

Pokud takový dokument nemáte, vynechte parametr -manifest *manifest.file* v příkazu **mt** .

Přizpůsobte příkazy v příkladech v produktu Obrázek 113 na stránce 923 nebo Obrázek 114 na stránce 923 za účelem kompilace a propojení `amqsaxe0.c` v systému Windows. Tyto příkazy pracují s Microsoft Visual Studio 2008, 2010 nebo 2012. Příklady předpokládají, že adresář `C:\Program Files\IBM\MQ\tools\c\samples` je aktuální adresář.

32bitová

```
cl /c /nologo /MD /Foamqsaxe0.obj amqsaxe0.c

link /nologo /dll /def:amqsaxe.def



amqsaxe0.obj \
   /manifest /out:amqsaxe.dll

mt -nologo -manifest amqsaxe.dll.manifest \
   -outputresource:amqsaxe.dll;2
```

Obrázek 113. Kompilace a odkaz `amqsaxe0.c` na 32bitovém serveru Windows

64 bitů

```
cl /c /nologo /MD /Foamqsaxe0.obj amqsaxe0.c

link /nologo /dll /def:amqsaxe.def \
     /libpath:..\..\lib64 \



amqsaxe0.obj /manifest /out:amqsaxe.dll

mt -nologo -manifest amqsaxe.dll.manifest \
   -outputresource:amqsaxe.dll;2
```

Obrázek 114. Kompilace a propojení `amqsaxe0.c` na 64bitovém serveru Windows

Související pojmy

"Ukázkový program uživatelské procedury rozhraní API" na stránce 1051
Ukázková uživatelská procedura rozhraní API vygeneruje trasování MQI do uživatelem určeného souboru s předponou definovanou v proměnné prostředí MQAPI_TRACE_LOGFILE.

zapIBM i
Kompilace uživatelských procedur rozhraní API na serveru IBM i.

Uživatelská procedura je vytvořena následujícím způsobem (v případě příkladu jazyka C):

1. Vytvořte modul pomocí příkazu CRTCMOD. Zkompilujte jej tak, aby používal teraspace tím, že zahrnete parametr TERASPACE(*YES *TSIFC).
2. Vytvořte servisní program z modulu pomocí CRTSRVPGM. Musíte ji svázat s servisním programem QMQM/LIBMQMZF_R pro ukončení rozhraní API s podporou podprocesů.

Konfigurace uživatelských procedur rozhraní API

Chcete-li povolit ukončení rozhraní API tím, že změníte informace o konfiguraci, nakonfigurujte produkt IBM MQ .

Chcete-li změnit informace o konfiguraci, musíte změnit stanzy, které definují uživatelské rutiny a pořadí, v jakém se spouštějí. Tyto informace lze změnit následujícími způsoby:

- Použití produktu IBM MQ Explorer (na platformách Windows a Linux (platformyx86 a x86-64))

- Použití příkazu **amqmdain** (v systému Windows)
- Přímé použití souborů mqs.ini a qm.ini (v systémech Windows, IBM i, UNIX and Linux).

  Soubor mqs.ini obsahuje informace vztahující se ke všem správcům front v konkrétním uzlu. Můžete jej najít v adresáři `/var/mqm` na UNIX and Linux IBM i, v adresáři `/QIBM/UserData/mqm` na IBM i a v WorkPath uvedeném v klíči `HKLM\SOFTWARE\IBM\WebSphere MQ` na systémech Windows .

  Soubor qm.ini obsahuje informace vztahující se ke specifickému správci front. Pro každého správce front je k dispozici jeden konfigurační soubor správce front, který je umístěn v kořenovém adresáři adresářového stromu obsazeného správcem front. Příklad: Cesta a název konfiguračního souboru pro správce front s názvem QMNAME je:

  V systémech UNIX and Linux:

  ```
  /var/mqm/qmgrs/QMNAME/qm.ini
  ```

  IBM i V systémech IBM i:

  ```
  /QIBM/UserData/mqm/qmgrs/QMNAME/qm.ini
  ```

  V systémech Windows:

  ```
  C:\ProgramData\IBM\MQ\qmgrs\QMNAME\qm.ini
  ```

  Před úpravou konfiguračního souboru jej zálohujte tak, abyste měli kopii, na kterou se můžete vrátit, pokud k tomu dojde.

  Konfigurační soubory můžete upravit buď:

  – Automaticky pomocí příkazů, které mění konfiguraci správců front v uzlu
  – Ruční při použití standardního textového editoru

  Pokud jste v atributu konfiguračního souboru nastavili nesprávnou hodnotu, hodnota se ignoruje a vydá se zpráva operátora, která daný problém označuje. (Efekt je stejný jako chybějící atribut zcela.)

Stanzy ke konfiguraci

Oddíly, které musí být změněny, jsou následující:

ApiExitCommon

Definováno v souboru mqs.ini a na stránce vlastností produktu IBM MQ Explorer na stránce vlastností produktu IBM MQ pod položkou Uživatelské procedury.

Při spuštění správce front jsou atributy v této sekci načteny a poté přepsané uživatelskými procedurami rozhraní API definovanými v souboru qm.ini.

ApiExitTemplate

Definováno v souboru mqs.ini a na stránce vlastností produktu IBM MQ Explorer na stránce vlastností produktu IBM MQ pod položkou Uživatelské procedury.

Je-li vytvořen správce front, budou atributy v této stanze zkopírovány do nově vytvořeného souboru qm.ini v lokální stanze ApiExit.

ApiExitLocal

Definováno v souboru qm.ini a na stránce IBM MQ Explorer na stránce vlastností správce front pod položkou Uživatelské procedury.

Při spuštění správce front jsou zde definované uživatelské procedury rozhraní API potlačují výchozí hodnoty definované v souboru mqs.ini.

Atributy pro stanzy

- Pojmenujte uživatelskou proceduru rozhraní API pomocí následujícího atributu:

  Název = ApiExit_name
  Popisný název uživatelské procedury rozhraní API předané do pole Název ExitInfostruktury MQAXP.

  Tento název musí být jedinečný, nesmí být delší než 48 znaků a smí obsahovat pouze platné znaky pro názvy objektů produktu IBM MQ (například názvy front).

- Identifikujte modul a vstupní bod kódu ukončení rozhraní API, které se mají spustit pomocí následujících atributů:

  Funkce=název_funkce
  Název vstupního bodu funkce do modulu, který obsahuje kód ukončení rozhraní API. Tento vstupní bod je funkce MQ_INIT_EXIT.

  Délka pole je omezena hodnotou MQ_EXIT_NAME_LENGTH.

  Modul = název_modulu
  Modul obsahující kód ukončení rozhraní API.

  Pokud pole obsahuje název modulu včetně úplné cesty, je použit beze změny.

  Pokud toto pole obsahuje pouze název modulu, je modul umístěn pomocí atributu `ExitsDefaultPath` v souboru `ExitPath` v souboru qm.ini.

  Na platformách, které podporují samostatné knihovny s podporou podprocesů, je třeba do modulu uživatelské procedury rozhraní API poskytovat jak nevláknovou, tak i verzi s podporou podprocesů. Svláknová verze musí mít příponu `_r` . Svláknová verze stubu aplikace IBM MQ implicitně připojuje `_r` k danému názvu modulu před jeho načtením.

  Délka tohoto pole je omezena na maximální délku cesty, kterou platforma podporuje.

- Volitelně předávejte data pomocí uživatelské procedury pomocí následujícího atributu:

  Data=název_dat
  Data, která mají být předána uživatelské proceduře rozhraní API, v poli ExitData struktury MQAXP.

  Pokud zahrnete tento atribut, úvodní a koncové mezery se odstraní, zbývající řetězec se ořízne na 32 znaků a výsledek se předá do ukončení. Pokud tento atribut vynecháte, bude pro ukončení předána výchozí hodnota 32 mezer.

  Maximální délka tohoto pole je 32 znaků.

- Identifikujte posloupnost tohoto ukončení ve vztahu k ostatním uživatelským procedurám pomocí následujícího atributu:

  Sequence=sequence_number
  Posloupnost, ve které je tato uživatelská procedura rozhraní API volána vzhledem k jiným uživatelským procedurám rozhraní API. Uživatelská procedura s nízkým pořadovým číslem se volá před ukončením s vyšším pořadovým číslem. Není třeba, aby pořadové číslování východů bylo souvislé. Posloupnost 1, 2, 3 má stejný výsledek jako posloupnost 7, 42, 1096. Pokud mají dvě uživatelské procedury stejné pořadové číslo, rozhodne správce front, který z nich má volat jako první. Můžete říci, které bylo voláno po události, tím, že jste čas nebo značku v oblasti ExitChainoznačené jako ExitChainAreaPtr v MQAXP nebo zápisem vašeho vlastního souboru protokolu.

  Tento atribut je nepodepsaná číselná hodnota.

Ukázkové stanzy

Ukázkový soubor mqs.ini obsahuje následující oddíly:

ApiExitTemplate
Tato stanza definuje ukončení s popisným názvem `OurPayrollQueueAuditor`, názvem modulu `auditora` pořadovým číslem 2. Datová hodnota 123 je předána uživatelské proceduře.

ApiExitCommon
Tato stanza definuje ukončení s popisným názvem `MQPoliceman`, názvem modulu `tmqp`a pořadovým číslem 1. Předané údaje jsou instrukce (`CheckEverything`).

```
mqs.ini

   ApiExitTemplate:
      Name=OurPayrollQueueAuditor
      Sequence=2
      Function=EntryPoint
      Module=/usr/ABC/auditor
      Data=123
   ApiExitCommon:
      Name=MQPoliceman
      Sequence=1
      Function=EntryPoint
      Module=/usr/MQPolice/tmqp
      Data=CheckEverything
```

Následující ukázkový soubor qm.ini obsahuje lokální definici ApiExitpro ukončení s popisným názvem `ClientApplicationAPIchecker`, názvem modulu `ClientAppChecker`a pořadovým číslem 3.

```
qm.ini

   ApiExitLocal:
      Name=ClientApplicationAPIchecker
      Sequence=3
      Function=EntryPoint
      Module=/usr/Dev/ClientAppChecker
      Data=9.20.176.20
```

Kanály-uživatelské programy pro kanály systému zpráv

Tato kolekce témat obsahuje informace o kanálových programech typu IBM MQ pro kanály systému zpráv.

Agenti kanálu zpráv (MCA) mohou také volat uživatelské procedury pro převod dat. Další informace o zápisu uživatelských procedur pro převod dat, viz “Zápis uživatelských procedur pro převod dat” na stránce 948.

Některé z těchto informací platí také pro uživatelské procedury na kanálech MQI, které se připojují k IBM MQ MQI clients ke správcům front. Další informace naleznete v tématu Programy ukončení kanálů pro kanály MQI.

Programy výstupních bodů kanálu jsou volány v definovaných místech zpracování prováděné programy MCA.

Některé z těchto programů uživatelských procedur pracují v komplementárních párech. Je-li například odesílající agent MCA volán odesílajícím programem MCA pro šifrování zpráv pro přenos, musí na konci procesu ukončit proces, který je komplementární, aby mohl být proces dekonstrukci ukončen.

Tabulka 127 na stránce 926 zobrazuje typy ukončení kanálu, které jsou dostupné pro každý typ kanálu.

Tabulka 127. Uživatelské procedury kanálu jsou dostupné pro každý typ kanálu.

Typ kanálu	Ukončení zprávy	Ukončení opakování zprávy	Ukončení příjmu	Uživatelská procedura pro zabezpečení zprávy	Ukončení odeslání	Uživatelská procedura automatické definice
Kanál odesílatele	Ano		Ano	Ano	Ano	
Kanál serveru	Ano		Ano	Ano	Ano	

Tabulka 127. Uživatelské procedury kanálu jsou dostupné pro každý typ kanálu. (pokračování)						
Typ kanálu	**Ukončení zprávy**	**Ukončení opakování zprávy**	**Ukončení příjmu**	**Uživatelská procedura pro zabezpečení zprávy**	**Ukončení odeslání**	**Uživatelská procedura automatické definice**
Kanál odesílatele klastru	Ano		Ano	Ano	Ano	Ano
Kanál příjemce	Ano	Ano	Ano	Ano	Ano	Ano
Kanál žadatele	Ano	Ano	Ano	Ano	Ano	
Přijímací kanál klastru	Ano	Ano	Ano	Ano	Ano	Ano
Kanál připojení klienta			Ano	Ano	Ano	
Kanál připojení serveru			Ano	Ano	Ano	Ano

Notes: z/OS

1. V systému z/OSse uživatelská procedura automatické definice používá pouze pro kanály odesílatele klastru a příjemce klastru.

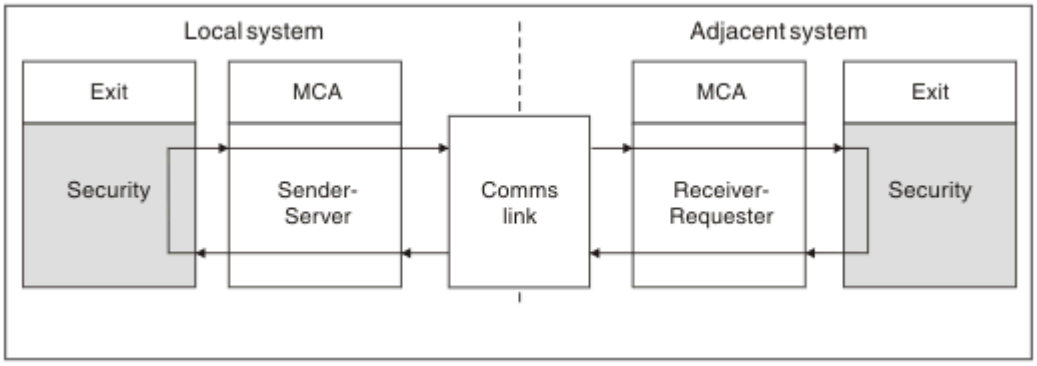
Chcete-li spustit uživatelské procedury kanálu na straně klienta, nelze použít proměnnou prostředí MQSERVER. Namísto toho vytvořte a vytvořte odkaz na tabulku definic kanálů klienta (CCDT), jak je popsáno v tématu Tabulka definic kanálů klienta.

Přehled zpracování

Přehled toho, jak MCAs používají programy uživatelské procedury kanálu.

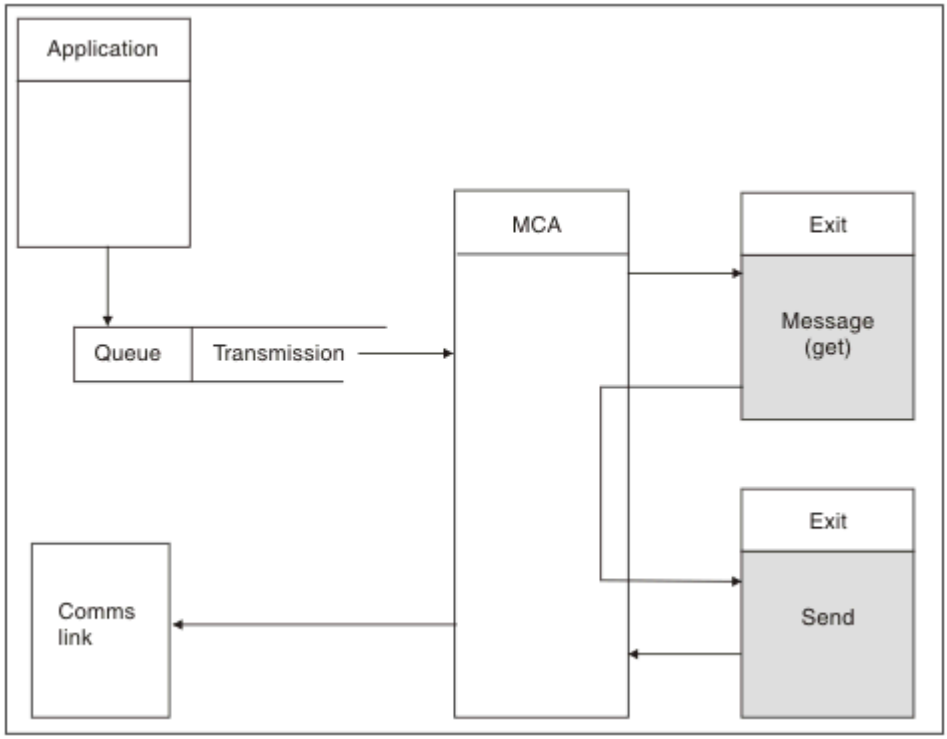
Při spuštění se při spuštění dialogového okna MMCA spustí dialogové okno spuštění synchronizace. Poté se přepnou na výměnu dat, která zahrnuje i uživatelské procedury zabezpečení. Tyto uživatelské procedury musí být úspěšně ukončeny pro dokončení fáze spuštění a aby bylo možné přenášet zprávy.

Fáze kontroly zabezpečení je smyčka, jak je zobrazeno v části Obrázek 115 na stránce 927.

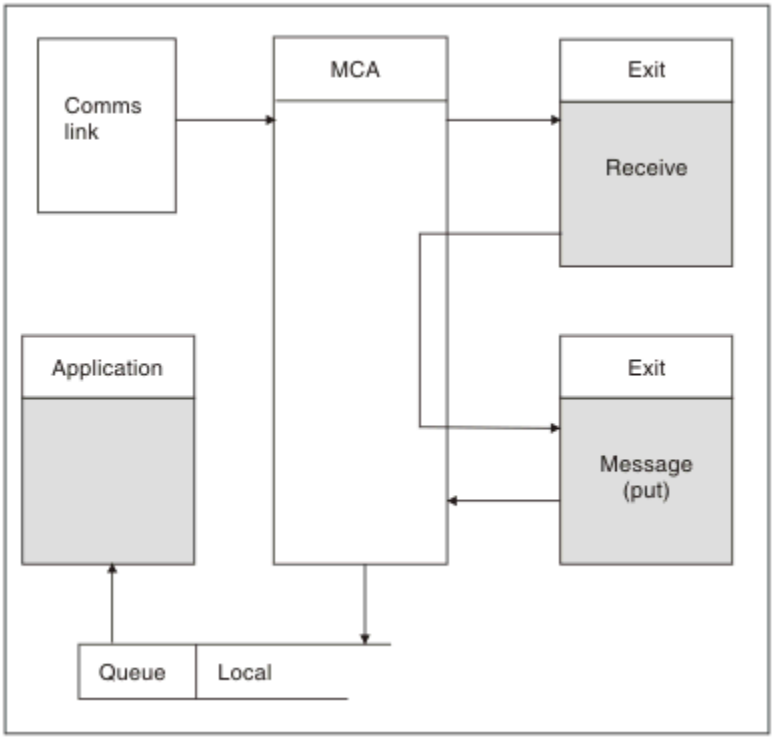


Obrázek 115. Smyčka ukončení zabezpečení

Během fáze přenosu zpráv odesílající agent MCA získává zprávy z přenosové fronty, volá uživatelskou proceduru zprávy, zavolá uživatelskou proceduru odeslání a odešle zprávu přijímajícímu programu MCA, jak je zobrazeno v části Obrázek 116 na stránce 928.



Obrázek 116. Příklad uživatelské procedury odeslání na konci kanálu zpráv odesílatele



Obrázek 117. Příklad uživatelské procedury příjmu na přijímacím konci kanálu zpráv

Přijímající agent MCA přijme zprávu z komunikačního spojení, zavolá uživatelskou proceduru pro přijetí zprávy, zavolá uživatelskou proceduru zprávy a pak umístí zprávu do lokální fronty, jak ukazuje Obrázek 117 na stránce 928. (Uživatelská procedura příjmu může být volána více než jednou, než se zavolá uživatelská procedura pro ukončení zprávy.)

Psaní programů výstupních bodů kanálu

Můžete použít následující informace, které vám pomohou psát programy výstupního bodu kanálu.

Uživatelské procedury a programy výstupního bodu kanálu mohou používat všechna volání MQI, s výjimkou těch, které jsou uvedeny v následujících oddílech. Pro produkt MQ V7 a novější obsahuje struktura MQCXP verze 7 a vyšší popisovač připojení hConn, který lze použít místo zadání volání MQCONN. Pro dřívější verze je nutné zadat příkaz MQCONN, i když je vráceno varování MQRC_ALREADY_CONNECTED, protože samotný kanál je připojen ke správci front.

Všimněte si, že uživatelská procedura kanálu musí zajišťovat neporušenost vláken.

U uživatelských procedur v kanálech připojení klienta závisí správce front, k němuž se pokus o připojení pokusí, závislá na tom, jak byla uživatelská procedura propojena. Pokud byla uživatelská procedura propojena s parametrem MQM.LIB (nebo QMQM/LIBMQM v systému IBM i) a nezadáte název správce front v rámci volání MQCONN, pokusí se uživatelská procedura připojit k výchozímu správci front v systému. Pokud byla uživatelská procedura propojena s parametrem MQM.LIB (nebo QMQM/LIBMQM v systému IBM i) a zadáte název správce front, který byl předán uživatelské proceduře prostřednictvím pole QMgrName objektu MQCD, se uživatelská procedura pokusí o připojení k tomuto správci front. Pokud byla uživatelská procedura propojena s MQIC.LIB nebo jakákoli jiná knihovna, volání MQCONN selže, ať již uvedete název správce front nebo ne.

Měli byste se vyhnout změně stavu transakce přidružené k předanému hConn ve výstupu kanálu; nesmíte použít příkazy MQCMIT, MQBACK nebo MQDISC s kanálem hConna nemůžete použít příkazové slovo MQBEGIN, které určuje kanál hConn.

Je-li MQCONNX použit při specifikaci MQCNO_HANDLE_SHARE_BLOCK nebo MQCNO_HANDLE_SHARE_NO_BLOCK k vytvoření nového připojení k produktu IBM MQ , je vaší zodpovědností zajistit správnou správu připojení a odpojení od správce front. Například uživatelská procedura kanálu, která vytváří nové připojení ke správci front při každém vyvolání bez odpojení, má za následek sestavení připojení a zvýšení počtu podprocesů agenta.

Ukončení se spouští ve stejném podprocesu jako samotný agent MCA a používá stejný popisovač připojení. Takže běží uvnitř stejné UOW jako MCA a všechna volání provedená v rámci synchronizačního bodu jsou potvrzována nebo vrácena kanálem na konci dávky.

Proto může uživatelská procedura kanálu pro zprávy odesílat zprávy s oznámením, které jsou potvrzeny pouze do této fronty, je-li potvrzena dávka obsahující původní zprávu. Je tedy možné vyslat volání MQI bodu synchronizace z uživatelské procedury pro zprávy kanálu.

Uživatelská procedura kanálu může měnit pole na disku MQCD. Tyto změny se však nepodnily, kromě případů uvedených v uvedených okolnostech. Pokud uživatelský program kanálu změní pole ve struktuře dat MQCD, bude nová hodnota procesem kanálu IBM MQ ignorována. Nová hodnota však zůstane na MQCD a je předávána všem zbývajícím uživatelským procedurám v řetězu ukončení a v libovolné konverzaci sdílející instanci kanálu. Další informace naleznete v tématu Změna polí MQCD v uživatelské proceduře kanálu

Také v případě programů napsaných v jazyku C nesmí být funkce knihovny non-reentrant jazyka C použita v programu ukončovacím programem kanálu.

Linux UNIX Pokud použijete více knihoven ukončení kanálu současně, mohou nastat problémy na některých platformách produktu UNIX and Linux , pokud kód pro dva různé uživatelské procedury obsahuje stejně pojmenované funkce. Je-li načtena uživatelská procedura kanálu, dynamický zavaděč v knihovně uživatelských procedur interpretuje názvy funkcí na adresy, na nichž je knihovna načtena. Pokud dvě uživatelské knihovny definují samostatné funkce, které mají identické názvy, může tento proces rozpoznání nesprávně interpretovat názvy funkcí jedné knihovny, aby používaly funkce jiné. Pokud se vyskytne tento problém, uveďte linker, že musí exportovat pouze požadované funkce exit a MQStart, protože tyto funkce nejsou ovlivněny. Jiné funkce musí mít lokální viditelnost, aby

se nepoužívaly pro funkce mimo vlastní knihovnu uživatelské procedury. Další informace naleznete v dokumentaci k řádku sestavovacího programu.

Všechny uživatelské procedury jsou volány spolu se strukturou parametrů uživatelské procedury kanálu (MQCXP), strukturou definice kanálu (MQCD), připravenou vyrovnávací pamětí dat, parametrem délky dat a parametrem délky vyrovnávací paměti. Délka vyrovnávací paměti nesmí být překročena:

- Pro ukončení zpráv je třeba povolit, aby největší zpráva byla odeslána přes kanál, a dále délka struktury MQXQH.
- Pro uživatelské procedury odeslání a přijetí je největší vyrovnávací paměť, kterou musíte povolit, takto:

  LU 6.2
  32 kB

  TCP:
  IBM i IBM i 16 kB

  IBM i Ostatní 32 kB

  Poznámka: Maximální použitelná délka může být o 2 bajty menší než tato délka. Pro podrobnosti zkontrolujte hodnotu vrácenou v MaxSegmentLength. Další informace o délce trvání MaxSegmentnaleznete v části MaxSegmentLength.

  NetBIOS:
  64 KB

  SPX:
  64 KB

  Poznámka: Uživatelské procedury příjmu odesílacích kanálů a uživatelské procedury odesílatele v přijímacích kanálech používají pro protokol TCP vyrovnávací paměti 2 kB.
- Pro uživatelské procedury zabezpečení přiděluje prostředek distribuované fronty vyrovnávací paměť o velikosti 4000 bajtů.

Je přípustné, aby východ vrátil náhradní pufr, spolu s příslušnými parametry. Podrobnosti o volání naleznete v příručce “Kanály-uživatelské programy pro kanály systému zpráv” na stránce 926 .

z/OS *Zápis ukončovacích programů kanálu v systému z/OS*

Můžete použít následující informace, které vám pomohou při psaní a kompilaci programů výstupních bodů kanálu pro produkt z/OS.

Uživatelské procedury jsou spuštěny jako z/OS LINK, v:

- Stav neautorizovaného problémového programu
- Řídicí režim primárního adresního prostoru
- Režim non-cross-memory
- Režim registru bez přístupu
- 31bitový režim adresování

Upravené moduly musí být umístěny v datové sadě určené příkazem CSQXLIB DD procedury adresního prostoru inicializátoru kanálu; názvy zaváděcích modulů jsou uvedeny jako názvy uživatelských procedur v definici kanálu.

Při vytváření výstupních procedur kanálu pro produkt z/OSplatí následující pravidla:

- Uživatelské procedury musí být napsány v assembleru nebo C; pokud se použije C, musí odpovídat programovacím prostředí systému C pro ukončení systému, popsané v příručce z/OS C/C++ Programming Guide.
- Uživatelské procedury jsou načteny z neautorizovaných knihoven definovaných příkazem CSQXLIB DD. Zadání CSQXLIB má DISP=SHR, uživatelské procedury mohou být aktualizovány, zatímco je spuštěn inicializátor kanálu. Nová verze se použije, když je kanál restartován.
- Východy musí být reentrantní a mohou být spuštěny kdekoli ve virtuálním úložišti.

- Ukončení musí resetovat prostředí, při návratu, na položku při vstupu.
- Opustí musí uvolnit jakékoli získané úložiště, nebo se musí ujistit, že je uvolněn následným vyvoláním ukončení.

  Pro úložiště, které má přetrvávat mezi vyvoláními, použijte službu z/OS STORAGE, nebo funkci knihovny 4kmalc pro programování programování C.

  Další informace o této funkci najdete v tématu 4kmalc() -- Allocate Page-Aligned Storage.
- Lze použít všechna volání MQI produktu IBM MQ s výjimkou MQCMIT nebo CSQBCMT a MQBACK nebo CSQBBAK. Musí být obsaženy po MQCONN (s prázdným názvem správce front). Pokud jsou tato volání použita, výstupní bod musí být upraven pomocí odkazu na stub CSQXSTUB.

  Výjimka z tohoto pravidla spočívá v tom, že uživatelské procedury kanálu zabezpečení mohou vydat volání pro potvrzení a odvolání MQI. Chcete-li tato volání vydat, uveďte příkazy CSQXCMT a CSQXBAK na místo MQCMIT nebo CSQBCMT a MQBACK nebo CSQBBAK.
- Všechny uživatelské procedury, které používají stub CSQXSTUB z produktu IBM WebSphere MQ 7.0 nebo novější, musí být linkové úpravy v zaváděcí knihovně CSQXLIB s formátem PDS-E.
- Ukončení nesmí používat žádné systémové služby, které by způsobily čekání, protože používání systémových služeb by mohlo výrazně ovlivnit zpracování některých nebo všech ostatních kanálů. Mnoho kanálů je obvykle spuštěno pod jednou TCB. Pokud uděláte něco v rámci procedury, která způsobí čekání a nepoužijete MQXWAIT, všechny tyto kanály se budou čekat. Causing channels to wait does not give any functional problems, but might have an negactional effect on performance. Většina SVCs zahrnuje čekání, takže se jim musíte vyhnout, s výjimkou následujících SVC:
  - GETMAIN/FREEMAIN/STORAGE
  - NAČÍST/ODSTRANIT

  Obecně proto, vyhýbejte se SVC, PC a I/O. Místo toho použijte volání MQXWAIT.
- Konce nevydávají ESTAE nebo SPIEs, kromě toho ve všech podúlohách, které se připojují, protože jejich ošetření chyb může kolidovat s ošetřením chyb prováděnými produktem IBM MQ. To znamená, že produkt IBM MQ nemusí být schopen provést zotavení z chyby nebo že váš uživatelský program nemusí přijímat všechny informace o chybě.
- Volání MQXWAIT (viz MQXWAIT). poskytuje službu čekání, která čeká na I/O a jiné události; je-li tato služba použita, uživatelské procedury nesmí používat zásobník sestavení.

  U I/O a dalších zařízení, která neposkytují neblokovaná zařízení nebo má ECB čekat, je třeba oddělit podúlohu a její dokončení čekalo na MQXWAIT; kvůli zpracování, které tato technika způsobuje, musí být tato služba použita pouze uživatelskou procedurou zabezpečení.

- Volání MQDISC MQI nezpůsobí výskyt implicitního potvrzení v rámci ukončovacího programu. Potvrzení procesu kanálu se provádí pouze v případě, že určuje protokol kanálu.

S produktem IBM MQ for z/OSjsou k dispozici následující ukázky uživatelských procedur:

CSQ4BAX0
Tato ukázka je napsána v asembleru a ilustruje použití rozhraní MQXWAIT.

CSQ4BCX1 a CSQ4BCX2
Tyto ukázky jsou napsány v jazyce C a ilustrují, jak získat přístup k parametrům.

CSQ4BCX3 a CSQ4BAX3
Tyto ukázky jsou napsány v jazyce C a asembleru.

Ukázka CSQ4BCX3 (která je předkompilovaná do SCSQAUTH LOADLIB, by měla fungovat bez jakýchkoli změn nezbytných na vlastní uživatelské proceduře). Můžete vytvořit LOADLIB (například s názvem MY.TEST.LOADLIB) a zkopírujte člen SCSQAUTH (CSQ4BCX3) do něj.

Chcete-li nastavit uživatelskou proceduru pro zabezpečení zprávy v připojení klienta, proveďte následující postup:

1. Navažte platný segment OMVS pro ID uživatele, které používá inicializátor kanálu.

To umožňuje inicializátoru kanálu IBM MQ for z/OS použít TCP/IP se soketovým rozhraním USS (UNIX System Services), aby se usnadnilo zpracování ukončení. Povšimněte si, že není nutné definovat segment OMVS pro ID uživatele všech připojovaných klientů.

2. Ujistěte se, že kód ukončení je sám o sobě spuštěn pouze v prostředí řízeném programem.

   To znamená, že vše načtené do adresního prostoru CHINIT musí být načteno z knihovny řízené programem (to znamená všechny knihovny v knihovně STEPLIB) a všechny knihovny pojmenované na CSQXLIB a

   ```
   ++hlq++.SCSQANLx
   ++hlq++.SCSQMVR1
   ++hlq++.SCSQAUTH
   ```

   Chcete-li nastavit zaváděcí knihovnu jako program kontrolovanou, použijte příkaz podobný tomuto příkladu:

   ```
   RALTER PROGRAM * ADDMEM('MY.TEST.LOADLIB'//NOPADCHK)
   ```

   Poté můžete programově aktivovat nebo aktualizovat prostředí řízené programem zadáním následujícího příkazu:

   ```
   SETROPTS WHEN(PROGRAM) REFRESH
   ```

3. Přidejte uživatelskou proceduru LOADLIB do třídy CSQXLIB DD (do procedury CHINIT started) zadáním následujícího příkazu:

   ```
   ALTER CHANNEL(xxxx) CHLTYPE(SVRCONN)SCYEXIT(CSQ4BCX3)
   ```

   Tím se aktivuje uživatelská procedura pro uvedený kanál.

4. Externí správce zabezpečení (ESM) uvádí všechny ostatní knihovny, které mají být řízeny programem, ale pamatujte, že žádné z knihoven ESM nebo C nemusí být pod kontrolou programu.

Další informace o nastavení uživatelské procedury zabezpečení s použitím ukázky CSQ4BCX3naleznete v tématu Kanál připojení serveruIBM MQ for z/OS .

CSQ4BCX4

Tato ukázka je napsána v jazyce C a demonstruje použití polí **RemoteProduct** a **RemoteVersion** v MQCXP.

Související pojmy

“Zápis ukončovacích programů kanálu v systému IBM i” na stránce 932
Můžete použít následující informace, které vám pomohou při psaní a kompilaci programů výstupních bodů kanálu pro produkt IBM i.

“Psaní ukončovacích programů kanálu v systému UNIX, Linux, and Windows” na stránce 933
Následující informace vám pomohou při psaní programů ukončovacích kanálů pro systémy UNIX, Linux, and Windows .

Související informace

IBM MQ for z/OS Kanál připojení serveru

IBM i *Zápis ukončovacích programů kanálu v systému IBM i*

Můžete použít následující informace, které vám pomohou při psaní a kompilaci programů výstupních bodů kanálu pro produkt IBM i.

Ukončení je programový objekt zapsaný v jazyce ILE C, ILE RPG nebo ILE COBOL. Názvy ukončovacích programů a jejich knihovny jsou pojmenovány v definici kanálu.

Při vytváření a kompilaci ukončovacího programu se řiďte následujícími podmínkami:

- Program musí být vytvořen jako bezpečný a vytvořen pomocí kompilátoru ILE C, ILE RPG nebo ILE COBOL. V případě ILE RPG musíte uvést specifikaci řízení THREAD (*SERIALIZE) a pro jazyk ILE COBOL musíte uvést SERIALIZE pro volbu THREAD příkazu PROCESS. Programy musí být také vázány na knihovny IBM MQ s podporou podprocesů: QMQM/LIBMQM_R v případě ILE C a ILE RPG, a AMQ0STUB_R v případě ILE COBOL. Další informace o tom, zda jsou aplikační podprocesy jazyka RPG nebo COBOL bezpečné, naleznete v příslušném programátorském průvodci pro jazyk.
- Produkt IBM MQ for IBM i vyžaduje, aby ukončovací programy byly aktivovány pro teraprostorová podpora. (Teraprostor je formou sdílené paměti zavedené v operačním systému OS/400 V4R4.) U kompilátorů ILE RPG a COBOL jsou všechny programy kompilované v systému OS/400 V4R4 nebo novější takto povoleny. Pro C, programy musí být kompilovány s volbami TERASPACE (*YES *TSIFC) uvedenými v příkazech CRTCMOD nebo CRTBNDC.
- Ukončení návratu ukazatele na vlastní vyrovnávací paměť musí zajistit, aby objekt ukazoval, že existuje mimo časové rozpětí programu uživatelské procedury kanálu. Ukazatel nemůže být adresou proměnné v zásobníku programů ani proměnné v haldě programu. Místo toho musí být ukazatel získán ze systému. Příklad je uživatelský prostor vytvořený v uživatelské proceduře. Chcete-li zajistit, aby každá datová oblast přidělená programem kanálu byla stále k dispozici pro program MCA při ukončení programu, musí být uživatelská procedura kanálu spuštěna v aktivační skupině volajícího nebo pojmenované aktivační skupiny. Proveďte to tak, že nastavíte parametr ACTGRP na CRTPGM na uživatelsky definovanou hodnotu nebo *CALLER. Je-li tento program vytvořen tímto způsobem, může program výstupního bodu kanálu přidělit dynamickou paměť a předat ukazatel této paměti zpět na agenta MCA.

Související pojmy

“Psaní ukončovacích programů kanálu v systému UNIX, Linux, and Windows” na stránce 933
Následující informace vám pomohou při psaní programů ukončovacích kanálů pro systémy UNIX, Linux, and Windows .

“Zápis ukončovacích programů kanálu v systému z/OS” na stránce 930
Můžete použít následující informace, které vám pomohou při psaní a kompilaci programů výstupních bodů kanálu pro produkt z/OS.

ULW *Psaní ukončovacích programů kanálu v systému UNIX, Linux, and Windows*

Následující informace vám pomohou při psaní programů ukončovacích kanálů pro systémy UNIX, Linux, and Windows .

Postupujte podle pokynů uvedených v tématu “Zapisování uživatelských procedur a instalovatelných služeb na UNIX, Linux a Windows” na stránce 898. Pokud je to vhodné, použijte následující informace specifické pro výstupní bod kanálu:

Uživatelská procedura musí být napsána v jazyce C a je to knihovna DLL v systému Windows.

Definujte v uživatelské proceduře fiktivní rutinu MQStart () a jako vstupní bod určete `MQStart` jako vstupní bod v knihovně. Obrázek 118 na stránce 933 ukazuje, jak nastavit položku pro váš program:

```
#include <cmqec.h>

void MQStart() {;} /* dummy entry point - for consistency only */
void MQENTRY ChannelExit ( PMQCXP  pChannelExitParms,
                           PMQCD   pChannelDefinition,
                           PMQLONG pDataLength,
                           PMQLONG pAgentBufferLength,
                           PMQVOID pAgentBuffer,
                           PMQLONG pExitBufferLength,
                           PMQPTR  pExitBufferAddr)
{
... Insert code here
}
```

Obrázek 118. Ukázkový zdrojový kód pro uživatelskou proceduru kanálu

Při zápisu kanálů pro produkt Windows s použitím jazyka Visual C + + je třeba vytvořit vlastní soubor DEF . Příklad toho, jak je ukázáno v Obrázek 119 na stránce 934. Další informace o zápisu ukončovacích programů kanálu najdete v tématu “Psaní programů výstupních bodů kanálu” na stránce 929.

```
EXPORTS
ChannelExit
```

Obrázek 119. Ukázkový soubor DEF pro Windows

Související pojmy

“Zápis ukončovacích programů kanálu v systému IBM i” na stránce 932
Můžete použít následující informace, které vám pomohou při psaní a kompilaci programů výstupních bodů kanálu pro produkt IBM i.

“Zápis ukončovacích programů kanálu v systému z/OS” na stránce 930
Můžete použít následující informace, které vám pomohou při psaní a kompilaci programů výstupních bodů kanálu pro produkt z/OS.

Ukončovací programy zabezpečení kanálu
Ukončovací programy zabezpečení můžete použít k ověření, že partner na druhém konci kanálu je pravý. To je známé jako ověření. Chcete-li určit, že kanál musí používat uživatelskou proceduru pro zabezpečení zprávy, zadejte do pole SCYEXIT definice kanálu název uživatelské procedury.

Poznámka: Ověření může být také dosaženo pomocí záznamů ověření kanálu. Záznamy ověření kanálu poskytují skvělou flexibilitu při prevenci přístupu ke správcům front z určitých uživatelů a kanálů a při mapování vzdálených uživatelů na identifikátory uživatelů produktu IBM MQ . Podpora TLS je také poskytována produktem IBM MQ k ověření vašich uživatelů a k poskytování kontrol šifrování a integrity dat pro vaše data. Další informace o protokolu TLS naleznete v tématu Protokoly zabezpečení TLS v produktu IBM MQ. Pokud však stále vyžadujete propracovanější (nebo odlišné) formy zpracování zabezpečení a další typy kontrol a zabezpečení kontextu zabezpečení, zvažte použití uživatelských procedur pro zabezpečení zápisu.

Pro uživatelské procedury zabezpečení zapsané před IBM WebSphere MQ 7.1 stojí za zmínku, že dřívější verze produktu IBM MQ dotazovali základního zabezpečeného soketu (např. GSKit) k určení rozlišujícího názvu partnera certifikátu vzdáleného partnera (SSLPEER) a rozlišovacího jména vydávajícího (SSLCERTI). V podpoře produktu IBM WebSphere MQ 7.1 byla přidána podpora pro rozsah nových atributů zabezpečení. Chcete-li získat přístup k těmto atributům, produkt IBM WebSphere MQ 7.1 získá kódování DER certifikátu a použije jej k určení DN subjektu a vydavatele. Atributy DN subjektu a vydavatele se zobrazují v následujících atributech stavu kanálu:

- SSLPEER (PCF selektor MQCACH_SSL_SHORT_PEER_NAME)
- SSLCERTI (PCF selector MQCACH_SSL_CERT_ISSUER_NAME)

Tyto hodnoty jsou vraceny příkazy pro stav kanálu a také data předaná uživatelským procedurám zabezpečení kanálu, jak je zobrazeno:

- MQCD SSLPeerNamePtr
- MQCXP SSLRemCertIssNamePtr

Atribut SERIALNUMBER v produktu IBM WebSphere MQ 7.1je také obsažen v DN subjektu a obsahuje sériové číslo pro certifikát vzdáleného partnera. Také některé atributy DN jsou vráceny v jiné posloupnosti z předchozích vydání. Následně se změní složení polí SSLPEER a SSLCERTI v produktu IBM WebSphere MQ 7.1 z předchozích verzí, a proto se doporučuje, aby všechny bezpečnostní východy nebo aplikace závislé na těchto polích byly zkontrolovány a aktualizovány.

Existující filtry názvů rovnocenných uzlů produktu IBM MQ zadané prostřednictvím pole SSLPEER definice kanálu nejsou ovlivněny a pokračují v práci stejným způsobem jako v předchozích verzích. Důvodem je to, že odpovídající algoritmus názvu partnera produktu IBM MQ byl aktualizován ke zpracování existujících filtrů SSLPEER, aniž by bylo nutné měnit definice kanálu. Tato změna s největší pravděpodobností ovlivní uživatelské procedury zabezpečení a aplikace, které závisejí na hodnotě DN subjektu a DN vydávajícího programového rozhraní PCF.

Uživatelská procedura zabezpečení může být zapsána v jazyce C nebo Java.

Ukončovací programy zabezpečení kanálu jsou volány na následujících místech v cyklu zpracování agenta MCA:

- Při inicializaci a ukončení programu MCA.
- Okamžitě po dokončení počátečního vyjednávání dat při spuštění kanálu. Přijímač nebo konec serveru kanálu může iniciovat výměnu zpráv o zabezpečení se vzdáleným koncem tím, že poskytuje zprávu, která má být doručena do ukončení zabezpečení na vzdáleném konci. Může se také stát, že k tomu bude úpadek. Ukončovací program se spustí znovu, aby zpracoval jakoukoli zprávu o zabezpečení přijatou ze vzdáleného ukončení.
- Okamžitě po dokončení počátečního vyjednávání dat při spuštění kanálu. Odesilatel nebo žadatel o ukončení kanálu zpracovává zprávu zabezpečení přijatou ze vzdáleného ukončení, nebo iniciuje výměnu zabezpečení, když vzdálený konec nemůže. Ukončovací program se spustí znovu, aby zpracoval všechny následné zprávy zabezpečení, které mohou být přijaty.

Žadatelský kanál se nikdy nevolá s rozhraním MQXR_INIT_SEC. Kanál oznamuje serveru, že má uživatelský program zabezpečení, a server pak má možnost zahájit uživatelskou proceduru pro zabezpečení zprávy. Pokud ji nemá, informuje o tom žadatele a tok s nulovou délkou se vrátí do výstupního programu.

Poznámka: Vyhněte se odesílání zpráv zabezpečení s nulovou délkou.

Příklady dat vyměněných zabezpečovacími programy jsou ilustrovány v číslech Obrázek 120 na stránce 936 až Obrázek 123 na stránce 938. Tyto příklady ukazují pořadí událostí, které se vyskytnou při ukončení bezpečnostní procedury zásobníku, a ukončení zabezpečení odesílatele. Postupné řádky v číslech představují plynutí času. V některých případech se události na příjemce a odesilateli nekorelují, a proto se mohou vyskytnout ve stejnou dobu nebo v různých časech. V jiných případech událost na jednom ukončovacím programu má za následek doplňkovou událost, která se vyskytne později v jiném ukončovacím programu. Například v Obrázek 120 na stránce 936:

1. Příjemce a odesilatel jsou vyvolány s MQXR_INIT, ale tato vyvolání nejsou korelovaná a mohou se proto vyskytnout ve stejnou dobu nebo v různých časech.
2. Přijímač je dále vyvolán s MQXR_INIT_SEC, ale vrací MQXCC_OK, které nevyžaduje žádnou doplňkovou událost u uživatelské procedury odesílatele.
3. Odesilatel je nyní vyvolán s MQXR_INIT_SEC. Tato hodnota není korelována s vyvoláním přijímače s MQXR_INIT_SEC. Odesilatel vrátí zprávu MQXCC_SEND_SEC_MSG, která způsobí doplňkovou událost při ukončení přijímače.
4. Příjemce se pak vyvolá s MQXR_SEC_MSG a vrátí MQXCC_SEND_SEC_MSG, což způsobí doplňkovou událost u uživatelské procedury pro odeslání zprávy.
5. Odesilatel je pak vyvolán s MQXR_SEC_MSG a vrací MQXCC_OK, což nevyžaduje žádnou doplňkovou událost na ukončení přijímače.

Receiver exit	Sender exit
Invoked with MQXR_INIT Responds with MQXCC_OK	Invoked with MQXR_INIT Responds with MQXCC_OK
Invoked with MQXR_INIT_SEC Responds with MQXCC_OK	
	Invoked with MQXR_INIT_SEC Responds with MQXCC_SEND_SEC_MSG
Invoked with MQXR_SEC_MSG Responds with MQXCC_SEND_SEC_MSG	
	Invoked with MQXR_SEC_MSG Responds with MQXCC_OK
Message transfer begins	

Obrázek 120. Výměna iniciovaná odesilatelem se smlouvou

Receiver exit	Sender exit
Invoked with MQXR_INIT Responds with MQXCC_OK	Invoked with MQXR_INIT Responds with MQXCC_OK
Invoked with MQXR_INIT_SEC Responds with MQXCC_OK	
	Invoked with MQXR_INIT_SEC Responds with MQXCC_SEND_SEC_MSG
Invoked with MQXR_SEC_MSG Responds with MQXCC_OK	
	Invoked with MQXR_SEC_MSG Responds with MQXCC_SUPPRESS_FUNCTION
Channel closes	
Invoked with MQXR_TERM Responds with MQXCC_OK	Invoked with MQXR_TERM Responds with MQXCC_OK

Obrázek 121. Výměna iniciovaná odesilatelem bez shody

Receiver exit	Sender exit
Invoked with MQXR_INIT Responds with MQXCC_OK	Invoked with MQXR_INIT Responds with MQXCC_OK
Invoked with MQXR_INIT_SEC Responds with MQXCC_SEND_SEC_MSG	
	Invoked with MQXR_SEC_MSG Responds with MQXCC_SEND_SEC_MSG
Invoked with MQXR_SEC_MSG Responds with MQXCC_OK	
Message transfer begins	
Invoked with MQXR_TERM Responds with MQXCC_OK	Invoked with MQXR_TERM Responds with MQXCC_OK

Obrázek 122. Výměnu zahájená příjemcem s dohodou

Receiver exit	Sender exit
Invoked with MQXR_INIT Responds with MQXCC_OK	Invoked with MQXR_INIT Responds with MQXCC_OK
Invoked with MQXR_INIT_SEC Responds with MQXCC_SEND_SEC_MSG	
	Invoked with MQXR_SEC_MSG Responds with MQXCC_OK
Invoked with MQXR_SEC_MSG Responds with MQXCC_SUPPRESS_FUNCTION	
Channel closes	

Obrázek 123. Výměnu přijímanou přijímačem bez dohody

Uživatelský program zabezpečení kanálu předává vyrovnávací paměť agenta obsahující data zabezpečení, kromě všech záhlaví přenosu generovaných uživatelskou procedurou pro zabezpečení zprávy. Tato data mohou být vhodná k tomu, aby bylo možné provést ověření zabezpečení buď na konci kanálu.

Uživatelský program zabezpečení na odesílajícím a přijímajícím na konci kanálu zpráv může vrátit buď dva kódy odezvy pro jakékoli volání:

- Výměna zabezpečení skončila bez chyb
- Potlačit kanál a zavřít jej

Poznámka:

1. Uživatelské procedury zabezpečení kanálu obvykle pracují ve dvojicích. Definujete-li vhodné kanály, ujistěte se, že jsou pro oba konce kanálu pojmenovány kompatibilní uživatelské programy.
2. IBM i V produktu IBM i mohou být programy ukončení zabezpečení, které byly zkompilovány s `Use adopted authority` (USEADPAUT = *YES), adoptovány oprávnění QMQM nebo QMQMADM. Je třeba dbát na to, aby tato uživatelská procedura nepoužívala tuto funkci k vytvoření bezpečnostního rizika pro váš systém.
3. Na kanálu TLS, na kterém druhý konec kanálu poskytuje certifikát, obdrží uživatelská procedura zabezpečení rozlišující název předmětu tohoto certifikátu v poli MQCD, ke kterému se přistupuje pomocí parametru SSLPeerNamePtr a rozlišující název vydavatele v poli MQCXP, ke kterému přistupuje SSLRemCertIssNamePtr. Používá se k tomu, který název může být umístěn:
   - Omezte přístup přes kanál TLS.
   - Chcete-li změnit MQCD.MCAUserIdentifier založený na názvu.

Související informace

Záznamy ověření kanálu

Koncepce zabezpečení přenosové vrstvy (TLS)

Psaní uživatelské procedury zabezpečení

Uživatelská procedura zabezpečení můžete zapsat pomocí kódu kostry ukončení zabezpečení.

Obrázek 124 na stránce 939 ilustruje, jak napsat uživatelskou proceduru zabezpečení.

```
void MQENTRY MQStart() {;}
void MQENTRY EntryPoint (PMQVOID pChannelExitParms,
                         PMQVOID pChannelDefinition,
                         PMQLONG pDataLength,
                         PMQLONG pAgentBufferLength,
                         PMQVOID pAgentBuffer,
                         PMQLONG pExitBufferLength,
                         PMQPTR  pExitBufferAddr)
{
  PMQCXP pParms = (PMQCXP)pChannelExitParms;
  PMQCD  pChDef = (PMQCD)pChannelDefinition;
  /* TODO: Add Security Exit Code Here */

}
```

Obrázek 124. Kód kostry uživatelské procedury zabezpečení

Standardní vstupní bod MQStart produktu IBM MQ musí existovat, ale nemusí provádět žádnou funkci. Název funkce (EntryPoint v tomto příkladu) může být změněn, ale funkce musí být exportována, když je knihovna kompilována a propojena. Stejně jako v předchozím příkladu musí být ukazatele pChannelExitParms přetypovány na PMQCXP a definice pChannelmusí být přetypovány na PMQCD. Všeobecné informace o ukončení volání kanálu a použití parametrů naleznete v tématu MQ_CHANNEL_EXIT. Tyto parametry se používají v proceduře zabezpečení následujícím způsobem:

PMQVOID pChannelExitParms
Vstup a výstup

Ukazatel na strukturu MQCXP-cast na PMQCXP pro přístup k polím. Tato struktura se používá ke komunikaci mezi uživatelskou procedurou a agentem MCA. Následující pole v aplikaci MQCXP mají zvláštní význam pro uživatelské procedury zabezpečení:

ExitReason
Sděluje zabezpečení Exit o aktuálním stavu v rámci výměny zabezpečení a používá se při rozhodování o tom, jaká akce se má provést.

ExitResponse
Odezva na agenta MCA, který diktuje další fázi v rámci výměny zabezpečení.

ExitResponse2
Přebytečné řídicí příznaky pro řízení způsobu, jakým program MCA interpretuje odezvu ukončení zabezpečení.

Oblast ExitUser
16 bajtů (maximum) úložiště, které může být použito uživatelskou procedurou zabezpečení k udržování stavu mezi voláními.

ExitData
Obsahuje data uvedená v poli SCYDATA definice kanálu (32 bajtů směrem doprava s mezerami).

Definice PMQVOID pChannel
Vstup a výstup

Ukazatel na strukturu MQCD-cast na PMQCD pro přístup k polím. Tento parametr obsahuje definici kanálu. Následující pole v produktu MQCD jsou zvláště zajímavá pro uživatelské procedury zabezpečení:

ChannelName
Název kanálu (20 bajtů směrem doprava s mezerami).

ChannelType
Kód definující typ kanálu.

Identifikátor uživatele MCA
Tato skupina tří polí je inicializována na hodnotu pole MCAUSER, která je uvedena v definici kanálu. Jakýkoli identifikátor uživatele uvedený v poli zabezpečení v těchto polích se používá pro řízení přístupu (nepoužitelné pro kanály SDR, SVR, CLNTCONN nebo CLUSSDR).

MCAUserIdentifier
Prvních 12 bajtů identifikátoru doplněného doprava o prázdné místo.

LongMCAUserIdPtr
Ukazatel na vyrovnávací paměť obsahující identifikátor plné délky (nezaručený nezajištěný null) má přednost před MCAUserIdentifier.

LongMCAUserIdLength
Délka řetězce, na kterou ukazuje LongMCAUserIdPtr -musí být nastaveno, je-li nastavena hodnota LongMCAUserIdPtr .

Identifikátor vzdáleného uživatele
Vztahuje se pouze na dvojice kanálů CLNTCONN/SVRCONN. Není-li definována žádná uživatelská procedura zabezpečení CLNTCONN, budou tato tři pole inicializována klientem MCA klienta, takže mohou obsahovat identifikátor uživatele z prostředí klienta, který může být použit procedurou zabezpečení SVRCONN pro ověření a při určování identifikátoru uživatele MCA. Je-li definována uživatelská procedura zabezpečení CLNTCONN, pak tato pole nejsou inicializována a lze ji nastavit pomocí uživatelské procedury zabezpečení CLNTCONN nebo lze zprávy zabezpečení použít k předání identifikátoru uživatele z klienta na server.

Identifikátor RemoteUser
Prvních 12 bajtů identifikátoru bylo na pravé straně doplněno mezerami.

LongRemoteUserIdPtr
Ukazatel na vyrovnávací paměť obsahující identifikátor plné délky (nezaručený nezajištěný null) má přednost před identifikátorem RemoteUserIdentifier.

LongRemoteUserIdDélka
Délka řetězce, na kterou ukazuje LongRemoteUserIdPtr-, musí být nastavena, je-li nastavena hodnota LongRemoteUserIdPtr.

Délka PMQLONG pData
Vstup a výstup

Ukazatel na MQLONG. Obsahuje délku jakékoli procedury zabezpečení obsažené v AgentBuffer při vyvolání procedury zabezpečení. Musí být nastaveno uživatelskou procedurou zabezpečení na délku jakékoli zprávy odesílané v AgentBuffer nebo ExitBuffer.

PMQLONG pAgentBufferLength
Vstup

Ukazatel na MQLONG. Délka dat obsažených v AgentBuffer při vyvolání uživatelské procedury zabezpečení.

Vyrovnávací paměť PMQVOID pAgent
Vstup a výstup

Při vyvolání uživatelské procedury zabezpečení tato zpráva odkazuje na jakoukoli zprávu odeslanou z uživatelské procedury pro ukončení práce. Má-li parametr ExitResponse2 ve struktuře MQCXP nastavený příznak MQXR2_USE_AGENT_BUFFER (výchozí), musí zabezpečení ukončit tento parametr tak, aby ukazoval na odesílaná data zpráv.

PMQLONG pExitBufferLength
Vstup a výstup

Ukazatel na MQLONG. Tento parametr je inicializován na 0 při prvním vyvolání uživatelské procedury zabezpečení a vrácená hodnota se udržuje mezi voláními zabezpečení během výměny zabezpečení.

PMQPTR pExitBufferAddr
Vstup a výstup

Tento parametr se inicializuje na ukazatel Null při prvním vyvolání procedury zabezpečení a vrácená hodnota se bude udržovat mezi voláními zabezpečení během výměny zabezpečení. Je-li příznak MQXR2_USE_EXIT_BUFFER nastaven ve struktuře ExitResponse2 ve struktuře MQCXP, musí bezpečnostní procedura nastavit tento parametr tak, aby ukazovala na odesílaná data zprávy.

Rozdíly v chování mezi uživatelskými procedurami zabezpečení definovanými v párech kanálů CLNTCONN/SVRCONN a dalších dvojicích kanálů
Uživatelské procedury zabezpečení mohou být definovány na všech typech kanálů. Chování uživatelských procedur zabezpečení definovaných ve dvojicích kanálů CLNTCONN/SVRCONN se však mírně liší od uživatelských procedur zabezpečení definovaných v jiných párech kanálu.

Ukončení zabezpečení na kanálu CLNTCONN může nastavit identifikátor vzdáleného uživatele v definici kanálu pro zpracování partnerským výstupem SVRCONN nebo pro autorizaci OAM, pokud není definována žádná uživatelská procedura zabezpečení SVRCONN a pole MCAUSER v SVRCONN není nastaveno.

Není-li definována žádná uživatelská procedura zabezpečení CLNTCONN, pak je identifikátor vzdáleného uživatele v definici kanálu nastaven na identifikátor uživatele z klientského prostředí (který může být prázdný) klientem MCA.

Výměna zabezpečení mezi ukončenými prvky zabezpečení definovaná na dvojici kanálu CLNTCONN a SVRCONN je úspěšně dokončena, když funkce zabezpečení SVRCONN vrátí hodnotu ExitResponse MQXCC_OK. Výměna zabezpečení mezi ostatními dvojicemi kanálů je úspěšně dokončena, když uživatelská procedura zabezpečení, která iniciovala výměnu, vrátila ExitResponse MQXCC_OK.

Avšak kód MQXCC_SEND_AND_REQUEST_SEC_MSG ExitResponse lze použít k vynucení pokračování zabezpečení: Pokud je ExitResponse MQXCC_SEND_REQUEST_REQUEST_SEC_MSG vrácena pomocí CLNTCONN nebo SVRCONN Security Exit, musí partner ukončit odpověď odesláním zprávy zabezpečení (ne MQXCC_OK nebo null response) nebo se kanál ukončí. V případě uživatelských procedur zabezpečení definovaných v jiných typech kanálů se položka ExitResponse MQXCC_OK vrátila jako odezva na objekt MQXCC_SEND_AND_REQUEST_SEC_MSG z objektu Security Exit pro pokračování v rámci výměny zabezpečení, jako kdyby byla vrácena odezva s hodnotou Null, a nikoli při ukončení kanálu.

Uživatelská procedura zabezpečení SSPI
Produkt IBM MQ for Windows poskytuje proceduru zabezpečení, která poskytuje ověření pro kanály produktu IBM MQ pomocí rozhraní SSPI (Security Services Programming Interface). SSPI poskytuje integrovaná bezpečnostní zařízení produktu Windows.

Tato uživatelská procedura pro zabezpečení ochrany dat je určena pro klienta produktu IBM MQ i pro server IBM MQ .

Balíky zabezpečení jsou načteny z adresáře security.dll nebo secur32.dll. Tyto knihovny DLL se dodávají spolu s operačním systémem.

Jednosměrné ověření je poskytnuto v produktu Windowspomocí ověřovacích služeb NTLM. Dvoucestné ověření je poskytnuto v produktu Windows 2000pomocí ověřovacích služeb Kerberos .

Uživatelský program zabezpečení je dodáván ve zdrojovém formátu a ve formátu objektu. Kód objektu můžete použít tak, jak je, nebo můžete použít zdrojový kód jako výchozí bod k vytvoření svých vlastních uživatelských programů. Další informace o použití objektu nebo zdrojového kódu uživatelské procedury zabezpečení SSPI viz “Použití uživatelské procedury zabezpečení SSPI v systému Windows” na stránce 1112 .

Výstupní programy pro odesílání a příjem kanálů
Pomocí uživatelských procedur pro odesílání a příjem můžete provádět úlohy, jako je komprese dat a dekomprimace. Můžete určit seznam programů uživatelských procedur pro odeslání a přijetí, které mají být spuštěny v posloupnosti.

Ukončovací programy pro odesílání a příjem kanálů jsou volány na následujících místech v cyklu zpracování agenta MCA:

- Ukončovací programy pro odesílání a příjem jsou volány pro inicializaci programu MCA a pro ukončení programu MCA.
- Uživatelský program odeslání je vyvolán na jednom nebo na druhém konci kanálu, v závislosti na tom, kdy je odeslán přenos pro jeden přenos zprávy, okamžitě před odesláním přenosu přes linku. Poznámka 4 vysvětluje, proč jsou uživatelské procedury k dispozici v obou směrech i v případě, že kanály zpráv odesílají zprávy pouze v jednom směru.
- Uživatelský program příjmu je vyvolán na jednom nebo na druhém konci kanálu, v závislosti na tom, kdy byl přijat přenos pro jeden přenos zpráv, okamžitě po přenosu z odkazu. Poznámka 4 vysvětluje, proč jsou uživatelské procedury k dispozici v obou směrech i v případě, že kanály zpráv odesílají zprávy pouze v jednom směru.

Pro jeden přenos zpráv může existovat mnoho přenosů a může existovat mnoho iterací pro odesílací a přijímací výstupní programy před tím, než se zpráva dostane na konec zprávy na přijímajícím konci.

Ukončovací programy pro odeslání a přijetí kanálu jsou předávány vyrovnávací paměti agenta obsahující data přenosu, která jsou odeslána nebo přijata z komunikačního spojení. V případě ukončovacích programů pro odeslání je prvních 8 bajtů vyrovnávací paměti vyhrazeno pro použití agentem MCA a nesmí být změněno. Pokud program vrátí jinou vyrovnávací paměť, pak tyto první 8 bajtů musí existovat v nové vyrovnávací paměti. Formát dat prezentovaných pro ukončovací programy není definován.

Dobrý kód odezvy musí být vrácen ukončovacími programy pro odeslání a přijetí. Jakákoli jiná odpověď způsobí nestandardní ukončení agenta MCA (nestandardní konec).

Poznámka: Nevystavujte volání MQGET, MQPUT nebo MQPUT1 v rámci synchronizačního bodu z uživatelské procedury odeslání nebo přijetí.

Poznámka:

1. Uživatelské procedury pro odesílání a příjem obvykle pracují ve dvojicích. Například ukončení odeslání může komprimovat data a ukončení příjmu je dekomprimován nebo uživatelská procedura pro odeslání může zašifrovat data a dešifrovat ji při ukončení příjmu. Definujete-li vhodné kanály, ujistěte se, že jsou pro oba konce kanálu pojmenovány kompatibilní uživatelské programy.
2. Je-li pro kanál povolena komprese, jsou tyto uživatelské procedury předávány komprimovanými daty.

3. Uživatelské procedury odeslání a příjmu kanálu mohou být volány pro segmenty zpráv jiné než pro data aplikací, například stavové zprávy. Nejsou volány během dialogového okna spuštění ani fáze kontroly zabezpečení.
4. Ačkoli kanály zpráv odesílají zprávy pouze v jednom směru, data kanálového řízení, jako je například prezenční signál a ukončení dávkového zpracování, procházejí oběma směry a tyto východy jsou k dispozici v obou směrech také. Avšak některé z počátečních datových toků pro spuštění kanálu jsou vyloučeny ze zpracování některou z uživatelských procedur.
5. Existují okolnosti, za kterých mohou být uživatelské procedury odeslání a přijetí vyvolány mimo pořadí, například pokud spouštíte řadu ukončovacích programů nebo pokud spouštíte také ukončení zabezpečení. Poté, když je procedura příjmu poprvé volána pro zpracování dat, může přijmout data, která neprošla přes odpovídající uživatelskou proceduru odeslání. Pokud uživatelská procedura pro příjem právě provedla operaci, například dekomprese, aniž byste nejprve zkontrolovali, zda je tato operace vyžadována, výsledky by byly neočekávané.

   Musíte kódovat své uživatelské procedury pro odesílání a příjem takovým způsobem, že procedura příjmu může zkontrolovat, zda data, která přijímá, byla zpracována pomocí příslušné uživatelské procedury odeslání. Doporučeným způsobem, jak to provést, je kódovat uživatelské programy tak, aby:

   - Uživatelská procedura pro odeslání nastaví hodnotu devátého bajtu dat na 0 a před provedením operace posune všechna data po 1 bajtu. (Prvních 8 bajtů je vyhrazeno pro použití agentem MCA.)
   - Pokud uživatelská procedura příjmu přijme data s hodnotou 0 v bajtu 9, bude vědět, že data pocházejí z uživatelské procedury odeslání. Odstraní 0, provádí komplementární operaci a přesouvá výsledná data zpět o 1 bajt.
   - Pokud uživatelská procedura příjmu přijme data, která mají něco jiného než 0 v bajtu 9, předpokládá, že uživatelská procedura odeslání nebyla spuštěna, a odešle data zpět volajícímu beze změny.

   Při použití uživatelských procedur zabezpečení je kanál ukončen uživatelskou procedurou zabezpečení a lze ji volat bez příslušné uživatelské procedury pro přijetí zprávy. Jednou z možností, jak tomuto problému předejít, je kódovat uživatelskou proceduru zabezpečení nastavením příznaku v souboru MQCD.SecurityUserData nebo MQCD.SendUserData, například, když se uživatelská procedura rozhodne ukončit kanál. Poté musí uživatelská procedura odeslání kontrolovat toto pole a zpracovat data pouze v případě, že příznak není nastaven. Tato kontrola zabrání zbytečnému pozměnění dat odesláním ukončení odeslání, a tím zabrání výskytu chyb konverze, které by mohly nastat v případě, že uživatelská procedura zabezpečení přijala změněná data.

Programy ukončení odeslání kanálu-rezervace prostoru
Pomocí uživatelských procedur pro odesílání a přijímání můžete transformovat data před přenosem. Uživatelské programy odeslání kanálu mohou přidávat vlastní data o transformaci vyhrazením prostoru v přenosové vyrovnávací paměti.

Tato data jsou zpracována ukončovacím programem příjmu a pak jsou odebrána z vyrovnávací paměti. Můžete například chtít šifrovat data a přidat klíč zabezpečení pro dešifrování.

Jak rezervovat prostor a použít jej

Je-li volaný uživatelský program volán k inicializaci, nastavte pole *ExitSpace* MQXCP na počet bajtů, které mají být rezervovány. Podrobnosti viz MQCXP . *ExitSpace* lze nastavit pouze během inicializace, tj. pokud má *ExitReason* hodnotu MQXR_INIT. Je-li uživatelská procedura pro odeslání vyvolána bezprostředně před přenosem a *ExitReason* je nastavena na hodnotu MQXR_XMIT, jsou bajty *ExitSpace* vyhrazeny v přenosové vyrovnávací paměti. Produkt *ExitSpace* není podporován v systému z/OS.

Uživatelská procedura odeslání nemusí používat veškerý rezervovaný prostor. Může použít méně než *ExitSpace* bajtů nebo, pokud není vyrovnávací paměť pro přenos zaplněna, může uživatelská procedura použít více než vyhrazenou částku. Při nastavení hodnoty *ExitSpace*je nutné ponechat alespoň 1 kB dat zprávy v přenosové vyrovnávací paměti. Výkon kanálu může být ovlivněn, je-li vyhrazen vyhrazený prostor pro velké množství dat.

Vyrovnávací paměť přenosu je obvykle 32Kb bajtů dlouhá. Pokud však kanál používá TLS, je velikost přenosové vyrovnávací paměti snížena na 15352 bajtů, aby se vešly do maximální délky záznamu definované RFC 6101 a související řady standardů TLS. Dalších 1024 bajtů je vyhrazeno pro použití produktem IBM MQ, takže maximální prostor vyrovnávací paměti pro přenos použitelný pro uživatelské procedury odeslání je 14,328 bajtů.

Co se děje na přijímajícím konci kanálu

Programy ukončení příjmu kanálu musí být nastaveny tak, aby byly kompatibilní s odpovídajícími uživatelskými procedurami odeslání. Uživatelské procedury příjmu musí znát počet bajtů ve vyhrazeném prostoru a musí v tomto prostoru odebrat data.

Hromadné ukončení odeslání

Můžete určit seznam programů uživatelských procedur pro odeslání a přijetí, které mají být spuštěny v posloupnosti. IBM MQ udržuje celkem pro prostor vyhrazený všemi uživatelskými procedurami odeslání. Tento celkový prostor musí ponechat alespoň 1 kB dat zprávy v přenosové vyrovnávací paměti.

Následující příklad ukazuje, jak je prostor přidělen pro tři uživatelské procedury odeslání, které byly volány za sebou:

1. Při volání pro inicializaci:
   - Odesílatel výstupu A rezervuje 1 KB.
   - Uživatelská procedura ukončení B rezervuje 2 kB.
   - Uživatelská procedura odeslání C rezervuje 3 kB.
2. Maximální velikost přenosu je 32 kB a uživatelská data jsou 5 kB dlouhá.
3. Ukončení A je voláno s 5 kB dat; je k dispozici až 27 kB, protože 5 KB je vyhrazeno pro uživatelské procedury B a C. Výstup A přidá 1 KB, množství, které je rezervované.
4. Ukončení B je voláno s 6 kB dat; je k dispozici až 29 kB, protože 3 KB je vyhrazeno pro ukončení C. Výstup B přidá 1 KB, méně než 2 kB, které je rezervované.
5. Výstup C je volán s 7 kB dat. K dispozici je až 32 kB. Exit C přidá 10K, více než 3 KB rezervované. Tato částka je platná, protože celkové množství dat, 17 KB, je menší než maximum 32 kB.

Maximální velikost přenosové vyrovnávací paměti pro kanál používající TLS je 15 352 bajtů, nejedná se o 32Kb. Důvodem je to, že základní přenosové segmenty zabezpečeného soketu jsou omezeny na 16Kb a část prostoru je požadována pro režii záznamu TLS. Dalších 1024 bajtů je vyhrazeno pro použití produktem IBM MQ, takže maximální prostor vyrovnávací paměti pro přenos použitelný pro uživatelské procedury odeslání je 14,328 bajtů.

Ukončovací programy zpráv kanálu
Můžete použít uživatelskou proceduru zprávy kanálu k provedení úloh, jako je šifrování na odkazu, ověření nebo nahrazení příchozích ID uživatelů, konverze dat zprávy, žurnálování a zpracování referenčních zpráv. Můžete zadat seznam ukončovacích programů pro zprávy, které mají být spuštěny v posloupnosti.

Ukončovací programy pro zprávy kanálu jsou volány na následujících místech v cyklu zpracování agenta MCA:

- Při inicializaci a ukončení MCA
- Okamžitě poté, co odesílající agent MCA vydal volání MQGET
- Před přijetím příkazu MCA pro příjem volání MQPUT

Uživatelská procedura pro zprávy je předána vyrovnávací paměti agenta obsahující záhlaví přenosové fronty MQXQH a textu zprávy aplikace načteného z fronty. Formát MQXQH je uveden v záhlaví MQXQH-Transmission-queue header.

Pokud použijete referenční zprávy (tj. zprávy, které obsahují pouze záhlaví odkazující na nějaký jiný objekt, který má být odeslán), výstupní bod zprávy rozpozná záhlaví, MQRMH. Identifikuje objekt, načte

jej jakýmkoli způsobem a připojí jej k záhlaví a předá jej do kanálu MCA pro přenos do přijímajícího agenta MCA. Na přijímajícím agentovi MCA rozpozná další uživatelská procedura zprávy, že tato zpráva je referenční zprávou, extrahuje objekt a předá záhlaví do cílové fronty. Chcete-li získat další informace o referenčních zprávách a některých ukázkových uživatelských procedurách, které je obsluhují, prohlédněte si téma “Referenční zprávy” na stránce 762 a “Spuštění ukázek referenční zprávy” na stránce 1080 .

Uživatelské procedury pro zprávy mohou vrátit následující odpovědi:

- Odešlete zprávu (příkaz GET exit). Je možné, že zpráva byla změněna uživatelskou procedurou. (Tato funkce vrací MQXCC_OK.)
- Vložte zprávu do fronty (PUT exit). Je možné, že zpráva byla změněna uživatelskou procedurou. (Tato funkce vrací MQXCC_OK.)
- Nezpracujte tuto zprávu. Zpráva se umístí do fronty nedoručených zpráv (nedoručená fronta zpráv) pomocí agenta MCA.
- Zavřete kanál.
- Chybný návratový kód, který způsobí, že agent MCA bude abnormálně ukončen.

Poznámka:

1. Uživatelské procedury pro zprávy jsou volány jednou pro každou převedenou úplnou zprávu i v případě, že je zpráva rozdělena na části.
2. Linux UNIX Pokud poskytnete uživatelskou proceduru na serveru UNIX nebo Linux, automatická konverze ID uživatele na malá písmena (popsáno zde) nebude fungovat.
3. Ukončení se spouští ve stejném podprocesu jako samotná MCA. Běží také uvnitř stejné pracovní jednotky (UOW) jako agent MCA, protože používá stejný popisovač připojení. Takže všechna volání provedená v rámci synchronizačního bodu jsou potvrzena nebo vrácena kanálem na konci dávky. Například jeden výstupní program zprávy kanálu může odesílat zprávy oznámení jinému a tyto zprávy jsou potvrzeny do fronty pouze tehdy, je-li potvrzena dávka obsahující původní zprávu.

   Z ukončovacího programu pro zprávy kanálu proto můžete volat volání MQI bodu synchronizace.

Převod zpráv mimo uživatelskou proceduru zprávy
Před voláním ukončení zprávy přijímá přijímající agent MCA některé převody na zprávu. Toto téma popisuje algoritmy používané k provádění převodů.

Která záhlaví se zpracovávají

Převodní rutina se spustí v MCA přijímače před voláním uživatelské procedury pro zpracování zprávy. Rutina konverze začíná hlavičkou MQXQH na začátku zprávy. Převodní rutina poté zpracuje zřetězená záhlaví, která následují za MQXQH, a v případě potřeby provádí konverzi. Zřetězená záhlaví mohou přesahovat odsazení obsažené v parametru HeaderLength dat MQCXP, která se předává do ukončení zprávy příjemce. Následující záhlaví jsou převedena na místo:

- MQXQH (název formátu " `MQXMIT` ")
- MQMD (toto záhlaví je součástí MQXQH a nemá žádný název formátu)
- MQMDE (jméno formátu " `MQHMDE` ")
- MQDH (název formátu " `MQHDIST` ")
- MQWIH (název formátu " `MQHWIH` ")

Následující záhlaví nejsou převedena, ale přešlápla na to, jak sběrnice MCA pokračuje v zpracování zřetězených záhlaví:

- MQDLH (název formátu " `MQDEAD` ")
- libovolná záhlaví s názvy formátů začínajících třemi znaky 'MQH' (například " `MQHRF` ") které nejsou jinak zmíněny

Způsob zpracování záhlaví

Parametr Formát každého záhlaví IBM MQ je čten agentem MCA. Parametr Formát je 8 bajtů v záhlaví, což je 8 jednobajtových znaků obsahujících název.

Agent MCA poté interpretuje data podle jednotlivých záhlaví jako typ pojmenovaného typu. Je-li Formát názvem typu záhlaví, který je vhodný pro převod dat produktu IBM MQ , je převeden. Pokud se jedná o jiný název označující data mimo produktMQ (například MQFMT_NONE nebo MQFMT_STRING), pak program MCA zastaví zpracování záhlaví.

Co je MQCXP HeaderLength?

Parametr HeaderLength v datech MQCXP dodávaném do uživatelské procedury pro zprávy je celková délka záhlaví MQXQH (která zahrnuje záhlaví MQMD), MQMDE a MQDH na začátku zprávy. Tato záhlaví jsou zřetězená s použitím názvů a délek 'Formát'.

MQWIHKM

Zřetězená záhlaví mohou přesahovat mimo HeaderLength do oblasti uživatelských dat. Záhlaví MQWIH, je-li přítomno, je jedno z následujících záhlaví, které se zobrazují za záhlaví HeaderLength.

Je-li záhlaví MQWIH v řetězcích záhlavích, je před voláním ukončení zprávy přijímače převedeno na místo.

Ukončovací program opakování zprávy kanálu
Uživatelská procedura pro opakování zprávy kanálu je volána, když pokus o otevření cílové fronty nebude úspěšný. Pomocí uživatelské procedury můžete určit, za jakých okolností se má opakovat pokus, kolikrát se má opakovat a jak často.

Tato uživatelská procedura je také volána na přijímajícím konci kanálu v rámci inicializace a ukončení MCA.

Uživatelská procedura pro opakování zpráv kanálu je předána vyrovnávací paměti agenta obsahující záhlaví přenosové fronty, MQXQH a text zprávy aplikace načtený z fronty. Formát MQXQH je uveden v části Přehled pro MQXQH.

Ukončení je vyvoláno pro všechny kódy příčiny; uživatelská procedura určí, pro jaké kódy příčiny chce agent MCA opakovat, pro kolikrát a v jakých intervalech. (Hodnota sady počtu opakování zprávy při definování kanálu je předána této uživatelské proceduře na disku MQCD, ale tato hodnota může tuto hodnotu ignorovat.)

Pole Počet MsgRetryv MQCXP se zvyšuje o jedničku při vyvolání procedury MCA při každém vyvolání uživatelské procedury a uživatelská procedura vrací buď hodnotu MQXCC_OK s dobou čekání obsaženou v poli Interval MsgRetryMQCXP nebo MQXCC_SUPPRESS_FUNCTION. Opakované pokusy pokračují donekonečna, dokud uživatelská procedura nevrátí funkci MQXCC_SUPPRESS_FUNCTION v poli ExitResponse MQCXP. Informace o akcích provedených agentem MCA pro tyto kódy dokončení naleznete v dokumentu MQCXP .

Pokud budou všechna opakování neúspěšná, bude zpráva zapsána do fronty nedoručených zpráv. Není-li k dispozici žádná fronta nedoručených zpráv, kanál se zastaví.

Pokud nedefinujete uživatelskou proceduru opakování zpráv pro kanál a dojde k selhání, které pravděpodobně bude dočasné, například MQRC_Q_FULL, agent MCA použije počet opakování zpráv a intervaly opakování zpráv nastavené při definování kanálu. Je-li selhání trvalejší povahy a nedefinujete-li uživatelský program, zpráva se zapíše do fronty nedoručených zpráv.

Ukončovací program automatické definice kanálu
Uživatelská procedura automatické definice kanálu může být použita, když je přijat požadavek na spuštění přijímacího kanálu nebo kanálu připojení serveru, ale neexistuje žádná definice pro tento kanál (ne pro IBM MQ for z/OS). Lze ji také volat na všech platformách pro kanály odesílatele klastru a příjemce klastru, aby bylo možné upravit úpravu definice pro instanci kanálu.

Uživatelská procedura automatické definice kanálu může být volána na všech platformách kromě z/OS , když je přijat požadavek na spuštění přijímacího kanálu nebo kanálu připojení serveru, ale neexistuje žádná definice kanálu. Můžete ji použít k úpravě zadané výchozí definice pro automaticky definovaného

příjemce připojení nebo kanálu připojení serveru, SYSTEM.AUTO.RECEIVERnebo SYSTEM.AUTO.SVRCON. Popis, jak lze definice kanálů vytvořit automaticky, najdete v tématu Příprava kanálů .

Uživatelská procedura automatické definice kanálu může být volána také při přijetí požadavku ke spuštění kanálu odesílatele klastru. Lze ji volat pro kanály odesílatele klastru a příjemce klastru, aby bylo možné upravit úpravu definice pro tuto instanci kanálu. V tomto případě se výstup také použije na IBM MQ for z/OS. Běžným použitím uživatelské procedury automatické definice kanálu je změna názvů ukončení platnosti zpráv (MSGEXIT, RCVEXIT, SCYEXIT a SENDEXIT), protože názvy uživatelských procedur mají různé formáty na různých platformách. Není-li zadána žádná uživatelská procedura automatické definice kanálu, bude při standardním chování v produktu z/OS prozkoumán distribuovaný název uživatelské procedury ve tvaru *[path]/libraryname(function)* a může obsahovat až osm znaků funkce, jsou-li přítomny, nebo názvy knihovny. V systému z/OSmusí výstupní program automatické definice kanálu změnit pole adresovaná prostřednictvím MsgExitPtr, MsgUserDataPtr, SendExitPtr, SendUserDataPtr, ReceiveExitPtr a ReceiveUserDataPtr, nikoli MsgExit, MsgUserData, SendExit, SendUserData, Samotné pole ReceiveExit a ReceiveUserData.

Další informace naleznete v tématu Práce s automaticky definovanými kanály.

Stejně jako u jiných uživatelských procedur kanálů je seznam parametrů:

```
MQ_CHANNEL_AUTO_DEF_EXIT (ChannelExitParms, ChannelDefinition)
```

`ChannelExitParms` jsou popsány v MQCXP. `ChannelDefinition` je popsán v MQCD.

Objekt MQCD obsahuje hodnoty, které jsou použity ve výchozí definici kanálu, pokud tyto hodnoty nebyly při ukončení změněny. Uživatelská procedura může upravovat pouze část těchto polí; viz MQ_CHANNEL_AUTO_DEFEXIT. Pokus o změnu jiných polí však nezpůsobí chybu.

Uživatelská procedura automatické definice kanálu vrací odpověď MQXCC_OK nebo MQXCC_SUPPRESS_FUNCTION. Není-li vrácena žádná z těchto odpovědí, program MCA pokračuje ve zpracování, jako by byl vrácen objekt MQXCC_SUPPRESS_FUNKCE. To znamená, že automatická definice je opuštěna, není vytvořena žádná nová definice kanálu a kanál nelze spustit.

Kompilace ukončovacích programů kanálu v systémech Windows, UNIX and Linux

Následující příklady vám pomohou při kompilaci ukončovacích programů kanálů pro systémy Windows, UNIX and Linux .

Windows

Windows

Příkaz kompilátoru a propojovacího programu pro programy výstupního bodu kanálu na serveru Windows:

```
cl.exe /Ic:\mqm\tools\c\include /nologo /c myexit.c
link.exe /nologo /dll myexit.obj /def:myexit.def /out:myexit.dll
```

Systémy UNIX and Linux

Linux UNIX

V těchto příkladech je `exit` název knihovny a `ChannelExit` je název funkce. V systému AIX se soubor exportu nazývá `exit.exp`. Tato jména se používají v definici kanálu k odkazu na výstupní program s použitím formátu popsaného v tématu Definice kanálu MQCD-kanálu. Viz také parametr MSGEXIT příkazu DEFINE CHANNEL .

Ukázkové příkazy kompilátoru a propojovacího programu pro kanál jsou ukončeny v produktu AIX:

```
$ xlc_r -q64 -e MQStart -bE:exit.exp -bM:SRE -o /var/mqm/exits64/exit
exit.c -I/usr/mqm/inc
```

Ukázkové příkazy kompilátoru a propojovacího programu pro uživatelské procedury kanálu jsou na serveru HP-UX

```
$ c89 +DD64 +z -c -D_HPUX_SOURCE -o exit.o exit.c -I/opt/mqm/inc
$ ld -b exit.o +ee MQStart +ee ChannelExit -o
/var/mqm/exits64/exit -L/usr/lib/pa20_64 -lpthread
$ rm exit.o
```

Ukázkové příkazy kompilátoru a propojovacího programu pro kanál-uživatelské procedury v systému Linux , kde je správce front 32 bitů:

```
$ gcc -shared -fPIC -o /var/mqm/exits/exit exit.c -I/opt/mqm/inc
```

Ukázkové příkazy kompilátoru a propojovacího programu pro kanál-uživatelské procedury v systému Linux , kde je správce front 64bitový:

```
$ gcc -m64 -shared -fPIC -o /var/mqm/exits64/exit exit.c -I/opt/mqm/inc
```

Ukázkové příkazy kompilátoru a propojovacího programu pro kanál jsou ukončeny v produktu Solaris:

```
$ cc -xarch=v9 -mt -G -o /var/mqm/exits64/exit exit.c -I/opt/mqm/inc
-R/usr/lib/64 -lsocket -lnsl -ldl
```

Na straně klienta lze použít 32bitovou nebo 64bitovou uživatelskou proceduru. Tato uživatelská procedura musí být propojena s parametrem mqic_r.

V systému AIXmusí být všechny funkce, které jsou volány produktem IBM MQ , exportovány. Ukázkový soubor exportu pro tento soubor make:

```
#
!channelExit
MQStart
```

Konfigurace uživatelských procedur kanálu

Chcete-li volat uživatelskou proceduru kanálu, je třeba ji pojmenovat v definici kanálu.

Uživatelské procedury kanálu musí být pojmenovány v definici kanálu. Toto pojmenování můžete provést, když poprvé definujete kanály, nebo můžete přidat informace později pomocí příkazu MQSC ALTER CHANNEL. Můžete také poskytnout názvy uživatelských procedur kanálu v datové struktuře kanálu MQCD. Formát názvu uživatelské procedury závisí na použité platformě IBM MQ . Informace naleznete v dokumentu MQCD nebo Commands (MQSC) Commands .

Pokud definice kanálu neobsahuje jméno programu uživatelské procedury, uživatelská procedura se nezavolá.

Uživatelská procedura automatické definice kanálu je vlastnost správce front, nikoli jednotlivým kanálem. Má-li být tato uživatelská procedura volána, musí být pojmenována v definici správce front. Chcete-li změnit definici správce front, použijte příkaz MQSC ALTER QMGR.

Zápis uživatelských procedur pro převod dat

Tato kolekce témat obsahuje informace o tom, jak psát uživatelské procedury pro zápis dat.

Poznámka: Nepodporováno v produktu MQSeries for VSE/ESA.

Provedete-li příkaz MQPUT, vytvoří aplikace deskriptor zprávy (MQMD) zprávy. Vzhledem k tomu, že produkt IBM MQ musí být schopen porozumět obsahu MQMD bez ohledu na platformu, na které je vytvořen, je systémem automaticky převeden.

Data aplikace však nejsou převedena automaticky. Pokud dochází k výměně znakových dat mezi platformami, kde se pole *CodedCharSetId* a *Encoding* liší, například mezi ASCII a EBCDIC, aplikace musí zajistit převod zprávy. Převod dat aplikací může provádět samotný správce front nebo uživatelský ukončovací program, který je označován jako *uživatelská procedura pro převod dat*. Správce front může provést vlastní převod dat pomocí jedné z vestavěných rutin pro převod, pokud se data aplikace nacházejí v jednom z vestavěných formátů (například MQFMT_STRING). Toto téma obsahuje informace o ukončovacím zařízení pro převod dat, který produkt IBM MQ poskytuje, když data aplikace nejsou ve vestavěném formátu.

Řízení může být předáno do uživatelské procedury pro převod dat během volání MQGET. Tím se vyhnete převodu mezi různými platformami dříve, než dosáhnete konečného cíle. Je-li však konečným cílem platforma, která nepodporuje převod dat na MQGET, musíte zadat CONVERT (YES) na odesílacím kanálu, který odesílá data do svého konečného cíle. Tím je zajištěno, že produkt IBM MQ převede data během přenosu. V takovém případě musí být váš výstup pro převod dat umístěn v systému, ve kterém je definován odesílací kanál.

Volání MQGET je vydáno přímo aplikací. Nastavte pole *CodedCharSetId* a *Encoding* v deskriptoru MQMD na požadovanou znakovou sadu a kódování. Pokud vaše aplikace používá stejnou znakovou sadu a kódování jako správce front, nastavte parametr *CodedCharSetId* na hodnotu MQCCSI_Q_MGR a *Encoding* na hodnotu MQENC_NATIVE. Po dokončení volání MQGET mají tato pole odpovídající hodnoty pro vrácená data zprávy. Tyto hodnoty se mohou lišit od hodnot požadovaných v případě, že převod nebyl úspěšný. Aplikace by měla tato pole resetovat na hodnoty vyžadované před každým voláním MQGET.

Podmínky vyžadované pro uživatelskou proceduru pro převod dat, které mají být volány, jsou definovány pro volání MQGET v MQGET.

Popis parametrů předaných ukončovacím programu pro převod dat a podrobných poznámek k použití naleznete v tématu Převod dat pro volání MQ_DATA_CONV_EXIT a strukturu MQDXP.

Programy, které převádějí data aplikací mezi různými kódováními a identifikátory CCSID, musí odpovídat rozhraní pro převod dat produktu IBM MQ (DCI).

V případě zavedení klientů výběrového vysílání je možné na straně klienta spustit uživatelské procedury rozhraní API a uživatelské procedury pro převod dat, protože některé zprávy nemusí procházet správcem front. Následující knihovny jsou nyní součástí balíků klienta stejně jako balíky serveru:

Tabulka 128. Knihovny, které jsou nyní v balících klienta a serveru

Operační systém	**Knihovny**
Windows	32 bit & 64 bit: mqm.dll & mqm.pdb
Linux & HP-UX	32 bit & 64 bit: libmqm.so & libmqm_r.so
AIX	32 bit & 64 bit: libmqm.a & libmqm_r.a
Solaris	32 bitů & 64 bitů: libmqm.so
IBM i	LIBMQM & LIBMQM_R

Vyvolání uživatelské procedury pro převod dat

Uživatelská procedura pro převod dat je uživatelská procedura, která přijímá řízení během zpracování volání MQGET.

Ukončení je vyvoláno, pokud jsou následující příkazy pravdivé:

- Volba MQGMO_CONVERT je určena v rámci volání MQGET.
- Některé nebo všechny z dat zprávy nejsou v požadované znakové sadě nebo kódování.
- Pole *Format* ve struktuře MQMD přidružené ke zprávě není MQFMT_NONE.
- Hodnota *BufferLength* zadaná v rámci volání MQGET není nulová.
- Délka dat zprávy není nula.

- Zpráva obsahuje data, která mají uživatelsky definovaný formát. Uživatelem definovaný formát může obsadit celou zprávu nebo může být předcházen jedním nebo více vestavěnými formáty. Například uživatelem definovaný formát může být před formátem MQFMT_DEAD_LETTER_HEADER. Ukončení je vyvoláno pro převod pouze uživatelem definovaného formátu; správce front převede všechny vestavěné formáty, které jsou před uživatelem definovaným formátem.

  Uživatelská procedura může být také vyvolána pro převod vestavěného formátu, ale k tomu dojde pouze v případě, že vestavěné převodní rutiny nemohou úspěšně převést vestavěný formát.

V poznámkách k použití volání MQ_DATA_CONV_EXIT ve skriptu MQ_DATA_CONV_EXITjsou popsány některé další podmínky.

Podrobnosti o volání MQGET najdete v tématu MQGET . Uživatelské procedury pro převod dat nemohou používat jiná volání MQI, než je MQXCNVC.

A new copy of the exit is loaded when an application attempts to retrieve the first message that uses that *Format* since the application connected to the queue manager. Je-li správce front vyřazen dříve načtenou kopií, může být také nová kopie načtena také v jiných časech.

Uživatelská procedura pro převod dat se spustí v prostředí, jako je tomu u programu, který vydal volání MQGET. Stejně jako uživatelské aplikace může být program MCA (agent kanálu zpráv) odesílající zprávy do cílového správce front, který nepodporuje převod zpráv. Prostředí zahrnuje adresní prostor a profil uživatele, kde je to vhodné. Uživatelská procedura nemůže ohrozit integritu správce front, protože není spuštěna v prostředí správce front.

Převod dat v systému z/OS

z/OS

V systému z/OSmějte na paměti následující skutečnosti:

- Ukončovací programy mohou být napsány pouze v sestavovacím jazyce.
- Ukončovací programy musí být reentrantní a mohou být spuštěny kdekoli v paměti.
- Ukončovací programy musí obnovit prostředí při ukončení na vstupu a musí uvolnit jakékoli získané úložiště.
- Ukončovací programy nesmí čekat ani vydat ESTAEs nebo SPIEs.
- Ukončovací programy jsou obvykle vyvolány, jako kdyby z/OS LINK in:
  - Stav neautorizovaného problémového programu
  - Řídicí režim primárního adresního prostoru
  - Režim bez přeběžení paměti
  - Režim bez přístupu-registr
  - 31bitový režim adresování
  - Režim TCB-PRB
- Při použití v aplikaci CICS je tato procedura vyvolána příkazem EXEC CICS LINK a musí odpovídat konvencím programování v produktu CICS . Parametry se předávají ukazateli (adresy) v komunikační oblasti CICS (COMMAREA).

  Přestože se nedoporučuje, uživatelské programy mohou používat volání rozhraní API produktu CICS také se zvýšenou opatrností:

  - Nevystavujte synchronizační body, protože výsledky by mohly ovlivnit jednotky práce deklarované agentem MCA.
  - Neaktualizujte žádné prostředky řízené správcem prostředků jiným než IBM MQ for z/OS, včetně těch, které jsou řízeny serverem CICS Transaction Server.

  V případě kanálů s hodnotou CONVERT = ANO je uživatelská procedura načtena z datové sady, na kterou odkazuje příkaz CSQXLIB DD. V produktu MQjsou uživatelské procedury CSQCBDCI a CSQCBDCO pro most IBM MQ CICS dodávány v SCSQAUTH.

Zápis ukončovacího programu pro převod dat pro IBM i

Informace o krocích, které je třeba zvážit při zápisu ukončovacích programů MQ pro převod dat v produktu IBM i.

Postupujte takto:

1. Pojmenujte svůj formát zprávy. Název se musí vejít do pole *Format* MQMD. Název *Format* nesmí obsahovat úvodní mezery a koncové mezery se budou ignorovat. Název objektu nesmí mít více než osm nemezerových znaků, protože *Format* je dlouhé pouze osm znaků. Nezapomeňte použít tento název pokaždé, když odešlete zprávu (náš příklad používá formát názvu).
2. Vytvořte strukturu pro znázornění vaší zprávy. Příklad naleznete v tématu Platná syntaxe .
3. Spusťte tuto strukturu přes příkaz CVTMQMDTA a vytvořte fragment kódu pro váš výstup konverze dat.

   Funkce generované příkazem CVTMMQMDTA používají makra, která jsou dodána v souboru QMQM/H (AMQSVMHA). Tato makra jsou zapsána za předpokladu, že jsou všechny struktury zabaleny; pozmění je, pokud tomu tak není.
4. Převeďte kopii dodaného zdrojového souboru kostry, QMQMSAMP/QCSRC (AMQSVFC4) a přejmenujte ji. (Náš příklad používá název EXIT_MOD.)
5. Najděte následující rámečky komentáře ve zdrojovém souboru a vložte kód podle popisu:

   a. Směrem ke konci zdrojového souboru, rámeček komentáře začíná znaky:

   ```
   /* Insert the functions produced by the data-conversion exit */
   ```

   Zde vložte fragment kódu vygenerovaný v kroku “3” na stránce 951.

   b. V blízkosti středu zdrojového souboru začíná rámeček s komentářem:

   ```
   /* Insert calls to the code fragments to convert the format's */
   ```

   Po této akci bude následovat komentář k funkci `ConverttagSTRUCT`.

   Změňte název funkce na název funkce, kterou jste přidali v kroku “5.a” na stránce 951. Chcete-li aktivovat funkci, odeberte znaky komentáře. Pokud existuje několik funkcí, vytvořte volání pro každou z nich.

   c. V blízkosti začátku zdrojového souboru začíná rámeček s komentářem:

   ```
   /* Insert the function prototypes for the functions produced by */
   ```

   Zde vložte příkazy prototypu funkce pro funkce přidané v kroku “5.a” na stránce 951.

   Pokud zpráva obsahuje znaková data, generuje vygenerovaný kód MQXCNVC; to lze vyřešit vytvořením vazby servisního programu QMQM/LIBMQM.
6. Zkompilujte zdrojový modul EXIT_MOD takto:

   ```
   CRTCMOD MODULE(library/EXIT_MOD) +
   SRCFILE(QCSRC) +
   TERASPACE(*YES *TSIFC)
   ```

7. Vytvořte/propojte program.

   Pro nevláknové aplikace použijte následující:

   ```
   CRTPGM PGM(library/Format)  +
      MODULE(library/EXIT_MOD)  +
      BNDSRVPGM(QMQM/LIBMQM)  +
      ACTGRP(QMQM)  +
      USRPRF(*USER)
   ```

Kromě vytvoření uživatelské procedury pro převod dat pro základní prostředí je v prostředí se závitem potřeba další. Tento zaveditelný objekt musí být následován _R. Knihovna LIBMQM_R slouží k interpretaci volání pro volání MQXCNVC. Oba zaveditelné objekty jsou požadovány pro prostředí s podprocesy.

```
CRTPGM PGM(library/Format_R)  +
   MODULE(library/EXIT_MOD)  +
   BNDSRVPGM(QMQM/LIBMQM_R)  +
   ACTGRP(QMQM)  +
   USRPRF(*USER)
```

8. Umístěte výstup do seznamu knihoven pro úlohu IBM MQ . Doporučuje se, aby v případě produkce byly výstupní programy pro převod dat uloženy v QSYS.

Poznámka:

1. Pokud CVTMQMDTA používá sbalené struktury, všechny aplikace IBM MQ musí použít kvalifikátor _Packed.
2. Ukončovací programy konverze dat musí být reentrantní.
3. MQXCNVC je jediné volání MQI, které lze vydat z uživatelské procedury pro převod dat.
4. Zkompilujte uživatelský program s volbou kompilátoru profilu uživatele nastaveným na *USER, aby byla uživatelská procedura spuštěna s oprávněním uživatele.
5. Povolení paměti teraprostoru je vyžadováno pro všechny uživatelské procedury s IBM MQ for IBM i ; Parametr TERASPACE (*YES *TSIFC) zadejte v příkazech CRTCMOD a CRTBNDC.

Zápis ukončovacího programu pro převod dat pro IBM MQ for z/OS

Informace o krocích, které je třeba vzít v úvahu při psaní ukončovacích programů pro převod dat pro produkt IBM MQ for z/OS.

Postupujte takto:

1. Vezměte dodanou zdrojovou kostru CSQ4BAX9 (pro jiná prostředí nežCICS) nebo CSQ4CAX9 (pro CICS) jako váš výchozí bod.
2. Spusťte obslužný program CSQUCVX.
3. Postupujte podle pokynů v prologu CSQ4BAX9 nebo CSQ4CAX9 k začlenění rutin generovaných obslužným programem CSQUCVX, a to v pořadí, ve kterém se struktury vyskytují ve zprávě, kterou chcete převést.
4. Obslužný program předpokládá, že datové struktury nejsou sbaleny, že je dodrženo implicitní zarovnání dat a že struktury začínají na celé hranici slova, s přeskočeným bajtem (jako mezi ID a VERZE v příkladu v Platné syntaxe). Pokud jsou struktury sbaleny, vynechte makra CMQXCALA, která jsou generována. Proto zvažte deklarování struktur takovým způsobem, že všechna pole jsou pojmenována a nejsou přeskočeny žádné bajty; v příkladu v části Platná syntaxepřidejte pole "MQBYTE DUMMY;" mezi ID a VERZE.
5. Dodaná uživatelská procedura vrací chybu, je-li vstupní vyrovnávací paměť kratší než formát zprávy, který má být převeden. Ačkoli uživatelská procedura převádí tolik úplných polí, jak je to možné, způsobí chybu vrácení nepřevedené zprávy do aplikace. Chcete-li povolit převod krátkých vstupních vyrovnávacích pamětí, pokud je to možné, včetně částečných polí, změňte hodnotu TRUNC= na makro CSQXCDFA na ANO: nevrátí se žádná chyba, takže aplikace obdrží převedenou zprávu. Aplikace musí ošetřit oříznutí.
6. Přidejte jakýkoli jiný speciální kód zpracování, který potřebujete.
7. Přejmenujte program na název formátu dat.
8. Kompilace a propojení-upravit program jako dávkový aplikační program (není-li určen pro použití s aplikacemi produktu CICS). Makra v kódu generovaném obslužným programem jsou v knihovně, **thlqual**.SCSQMACS.

   Pokud zpráva obsahuje znaková data, generuje vygenerovaný kód MQXCNVC. Pokud vaše uživatelská procedura používá toto volání, propojte ji s programem výstupního stubu CSQASTUB. Stub je nezávislý

na jazykovém prostředí a nezávislý na prostředí. Alternativně můžete stub načíst dynamicky s použitím dynamického názvu volání CSQXCNVC. Další informace viz “Dynamické volání stubu IBM MQ” na stránce 1007.

Umístěte modul pro úpravy odkazů do knihovny zátěže aplikace a do datové sady, na kterou odkazuje příkaz CSQXLIB DD vaší úlohy spuštěné vaším inicializačním programem kanálu.

9. Je-li uživatelská procedura určena pro použití aplikacemi produktu CICS , kompilací a linkem-upravte ji jako aplikační program produktu CICS , případně včetně CSQASTUB, je-li to vyžadováno. Umístěte jej do knihovny aplikačního programu CICS . Definujte program pro CICS obvyklým způsobem, uveďte EXECKEY (CICS) v definici.

Poznámka: Ačkoli jsou běhové knihovny LE/370 potřebné pro spuštění obslužného programu CSQUCVX (viz krok “2” na stránce 952), nejsou zapotřebí pro linkování nebo spuštění samotného ukončení konverze dat (viz kroky “8” na stránce 952 a “9” na stránce 953).

Informace o převodu dat v rámci mostu IBM MQ - IMS naleznete v příručce “Zápis aplikací mostu IMS” na stránce 62 .

Psaní uživatelské procedury pro převod dat pro produkt IBM MQ v systémech UNIX and Linux

Informace o krocích, které je třeba vzít v úvahu při zápisu ukončovacích programů pro převod dat pro produkt IBM MQ v systémech UNIX and Linux .

Postupujte takto:

1. Pojmenujte svůj formát zprávy. Název se musí vejít do pole *Format* deskriptoru MQMD a musí být zadán velkými písmeny, například MYFORMAT. Název *Format* nesmí obsahovat úvodní mezery. Koncové mezery jsou ignorovány. Název objektu nesmí mít více než osm nemezerových znaků, protože *Format* je dlouhé pouze osm znaků. Nezapomeňte použít tento název pokaždé, když odešlete zprávu.

   Je-li uživatelská procedura pro převod dat použita v prostředí s podprocesy, musí za něj následovat objekt loadable, aby indikoval, že se jedná o verzi s podporou podprocesů.
2. Vytvořte strukturu pro znázornění vaší zprávy. Příklad naleznete v tématu Platná syntaxe .
3. Spusťte tuto strukturu pomocí příkazu `crtmqcvx` a vytvořte fragment kódu pro váš výstup pro převod dat.

   Funkce generované příkazem `crtmqcvx` používají makra, která předpokládají, že všechny struktury jsou sbalené; jejich změnu, pokud se nejedná o tento případ.
4. Zkopírujte dodaný zdrojový soubor kostry, přejmenujte jej na název vašeho formátu zprávy, který jste nastavili v kroku “1” na stránce 953. Zdrojový soubor kostry a kopie jsou určeny pouze ke čtení.

   Zdrojový soubor kostry se nazývá amqsvfc0.c.
5. V systému IBM MQ for AIXje také dodán soubor exportu kostry s názvem amqsvfc.exp . Okopírujte tento soubor, přejmenujte jej na MYFORMAT.EXP.
6. Kostra obsahuje ukázkový soubor záhlaví, amqsvmha.h, v adresáři *MQ_INSTALLATION_PATH*/inc, kde *MQ_INSTALLATION_PATH* představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ . Ujistěte se, že vaše cesta Include ukazuje na tento adresář k vyzvednutí tohoto souboru.

   Soubor amqsvmha.h obsahuje makra, která jsou použita kódem generovaným příkazem `crtmqcvx` . Pokud struktura, která má být převedená, obsahuje znaková data, tato makra volají aplikaci MQXCNVC.
7. Najděte následující rámečky komentáře ve zdrojovém souboru a vložte kód podle popisu:
   a. Směrem ke konci zdrojového souboru, rámeček komentáře začíná znaky:

   ```
   /* Insert the functions produced by the data-conversion exit */
   ```

   Zde vložte fragment kódu vygenerovaný v kroku “3” na stránce 953.
   b. V blízkosti středu zdrojového souboru začíná rámeček s komentářem:

```
/* Insert calls to the code fragments to convert the format's */
```

Po této akci bude následovat komentář k funkci `ConverttagSTRUCT`.

Změňte název funkce na název funkce, kterou jste přidali v kroku “7.a” na stránce 953. Chcete-li aktivovat funkci, odeberte znaky komentáře. Pokud existuje několik funkcí, vytvořte volání pro každou z nich.

c. V blízkosti začátku zdrojového souboru začíná rámeček s komentářem:

```
/* Insert the function prototypes for the functions produced by */
```

Zde vložte příkazy prototypu funkce pro funkce přidané v kroku “3” na stránce 953.

8. Zkompilujte svou uživatelskou proceduru jako sdílenou knihovnu s použitím volby MQStart jako vstupního bodu. Chcete-li to provést, viz “Kompilování dat pro převod dat na systémech UNIX and Linux” na stránce 954.
9. Umístěte výstup do výstupního adresáře. Výchozí výstupní adresář je `/var/mqm/exits` pro 32 bitové systémy a `/var/mqm/exits64`pro 64bitové systémy. Tyto adresáře můžete změnit v souboru qm.ini nebo mqclient.ini . Tato cesta může být nastavena pro každého správce front a uživatelská procedura je vyhledána pouze v této cestě nebo v cestách.

Poznámka:

1. Pokud produkt `crtmqcvx` používá zabalené struktury, musí být tímto způsobem kompilovány všechny aplikace produktu IBM MQ .
2. Ukončovací programy konverze dat musí být reentrantní.
3. MQXCNVC je jediné volání MQI, které lze vydat z uživatelské procedury pro převod dat.

Kompilování dat pro převod dat na systémech UNIX and Linux
Příklady způsobu kompilace ukončení převodu dat na systémech UNIX and Linux .

Na všech platformách je vstupní bod do modulu MQStart.

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

AIX

Zkompilujte zdrojový kód ukončení zadáním jednoho z následujících příkazů:

32bitové aplikace

Nevláknová

```
cc -e MQStart -bE:MYFORMAT.exp -bM:SRE -o /var/mqm/exits/MYFORMAT \
     MYFORMAT.c -I  MQ_INSTALLATION_PATH/inc
```

Vláknové

```
xlc_r -e MQStart -bE:MYFORMAT.exp -bM:SRE -o /var/mqm/exits/MYFORMAT_r \
     MYFORMAT.c -I  MQ_INSTALLATION_PATH/inc
```

64bitové aplikace

Nevláknová

```
cc -q64 -e MQStart -bE:MYFORMAT.exp -bM:SRE -o /var/mqm/exits64/MYFORMAT \
     MYFORMAT.c -I MQ_INSTALLATION_PATH/inc
```

Vláknové

```
xlc_r -q64 -e MQStart -bE:MYFORMAT.exp -bM:SRE -o /var/mqm/exits64/MYFORMAT_r \
     MYFORMAT.c -I  MQ_INSTALLATION_PATH/inc
```

Platforma HP-UX Itanium

Zkompilujte a propojte zdrojový kód zadáním jedné z následujících sad příkazů:

32bitové aplikace

Nevláknová

Kompilace zdrojového kódu ukončení:

```
c89 +e +z -c -D_HPUX_SOURCE -o MYFORMAT.o MYFORMAT.c -I MQ_INSTALLATION_PATH/inc
```

Propojte objekt uživatelské procedury:

```
ld +b: -b MYFORMAT.o +ee MQStart -o \
    /var/mqm/exits/MYFORMAT -L/usr/lib/hpux32
rm MYFORMAT.o
```

Vláknové

Kompilace zdrojového kódu ukončení:

```
c89 +e +z -c -D_HPUX_SOURCE -o MYFORMAT.o MYFORMAT.c -I MQ_INSTALLATION_PATH/inc
```

Propojte objekt uživatelské procedury:

```
ld +b: -b MYFORMAT.o +ee MQStart -o \
    /var/mqm/exits/MYFORMAT_r -L/usr/lib/hpux32 \
    -lpthread
rm MYFORMAT.o
```

64bitové aplikace

Nevláknová

Kompilace zdrojového kódu ukončení:

```
 c89 +DD64 +e +z -c -D_HPUX_SOURCE -o MYFORMAT.o MYFORMAT.c -I MQ_INSTALLATION_PATH/inc
```

Propojte objekt uživatelské procedury:

```
ld -b MYFORMAT.o +ee MQStart \
    -o  /var/mqm/exits64/MYFORMAT \
    -L/usr/lib/hpux64
rm MYFORMAT.o
```

Vláknové

Kompilace zdrojového kódu ukončení:

```
c89 +DD64 +e +z -c -D_HPUX_SOURCE -o MYFORMAT.o MYFORMAT.c -I MQ_INSTALLATION_PATH/inc
```

Propojte objekt uživatelské procedury:

```
ld -b MYFORMAT.o +ee MQStart  \
```

```
    -o  /var/mqm/exits64/MYFORMAT_r  \
    -L/usr/lib/hpux64 -lpthread
rm MYFORMAT.o
```

Linux

Zkompilujte zdrojový kód ukončení zadáním jednoho z následujících příkazů:

31bitové aplikace

Nevláknová

```
gcc -m31 -shared -fPIC -o /var/mqm/exits/MYFORMAT MYFORMAT.c \
     -I  MQ_INSTALLATION_PATH/inc
```

Vláknové

```
gcc -m31 -shared -fPIC -o /var/mqm/exits/MYFORMAT_r MYFORMAT.c
     -I  MQ_INSTALLATION_PATH/inc
```

32bitové aplikace

Nevláknová

```
gcc -m32 -shared -fPIC -o /var/mqm/exits/MYFORMAT MYFORMAT.c
     -I  MQ_INSTALLATION_PATH/inc
```

Vláknové

```
gcc -m32 -shared -fPIC -o /var/mqm/exits/MYFORMAT_r MYFORMAT.c
     -I  MQ_INSTALLATION_PATH/inc
```

64bitové aplikace

Nevláknová

```
gcc -m64 -shared -fPIC -o /var/mqm/exits64/MYFORMAT MYFORMAT.c
     -I  MQ_INSTALLATION_PATH/inc
```

Vláknové

```
gcc -m64 -shared -fPIC -o /var/mqm/exits64/MYFORMAT_r MYFORMAT.c
     -I  MQ_INSTALLATION_PATH/inc
```

Solaris

Zkompilujte zdrojový kód ukončení zadáním jednoho z následujících příkazů:

32bitové aplikace

Platforma SPARC

```
cc -xarch=v8plus -KPIC -mt -G -o /var/mqm/exits/MYFORMAT \
   MYFORMAT.c -I MQ_INSTALLATION_PATH/inc -R/usr/lib/32 -lsocket -lnsl -ldl
```

Platformax86-64

```
cc -xarch=386 -KPIC -mt -G -o /var/mqm/exits/MYFORMAT \
   MYFORMAT.c -I MQ_INSTALLATION_PATH/inc -R/usr/lib/32 -lsocket -lnsl -ldl
```

64bitové aplikace

Platforma SPARC

```
cc -xarch=v9 -KPIC -mt -G -o /var/mqm/exits64/MYFORMAT \
   MYFORMAT.c -I MQ_INSTALLATION_PATH/inc -R/usr/lib/64 -lsocket -lnsl -ldl
```

Platformax86-64

```
cc -xarch=amd64 -KPIC -mt -G -o /var/mqm/exits64/MYFORMAT \
   MYFORMAT.c -I MQ_INSTALLATION_PATH/inc -R/usr/lib/64 -lsocket -lnsl -ldl
```

Psaní uživatelské procedury pro převod dat pro IBM MQ for Windows

Informace o krocích, které je třeba vzít v úvahu při psaní ukončovacích programů pro převod dat pro produkt IBM MQ for Windows.

Postupujte takto:

1. Pojmenujte svůj formát zprávy. Název se musí vejít do pole *Format* MQMD. Název *Format* nesmí obsahovat úvodní mezery. Koncové mezery jsou ignorovány. Název objektu nesmí mít více než osm nemezerových znaků, protože *Format* je dlouhé pouze osm znaků.

   Soubor .DEF s názvem amqsvfcn.def je také dodán v adresáři ukázek, *MQ_INSTALLATION_PATH*\Tools\C\Samples. *MQ_INSTALLATION_PATH* je adresář, kde je nainstalován produkt IBM MQ . Vezměte kopii tohoto souboru a přejmenujte ji, například na MYFORMAT.DEF. Ujistěte se, že název knihovny DLL, která se vytvoří, a jméno uvedené v souboru MYFORMAT.DEF jsou stejné. Přepište název FORMAT1 v souboru MYFORMAT.DEF s novým názvem formátu.

   Nezapomeňte použít tento název pokaždé, když odešlete zprávu.
2. Vytvořte strukturu pro znázornění vaší zprávy. Příklad naleznete v tématu Platná syntaxe .
3. Spusťte tuto strukturu pomocí příkazu `crtmqcvx` a vytvořte fragment kódu pro váš výstup pro převod dat.

   Funkce generované příkazem CRTMQCVX používají makra, která jsou zapsána za předpokladu, že jsou všechny struktury sbaleny; jejich změny, pokud se nejedná o tento případ.
4. Zkopírujte dodaný zdrojový soubor kostry, amqsvfc0.c, přejmenujte jej na název vašeho formátu zprávy, který jste nastavili v kroku “1” na stránce 957.

   amqsvfc0.c je v *MQ_INSTALLATION_PATH*\Tools\C\Samples , kde *MQ_INSTALLATION_PATH* je adresář, kde je nainstalován produkt IBM MQ . (Výchozí instalační adresář je `C:\Program Files\IBM\MQ`.)

   Kostra obsahuje ukázkový soubor záhlaví amqsvmha.h v adresáři *MQ_INSTALLATION_PATH*\Tools\C\include . Ujistěte se, že vaše cesta Include ukazuje na tento adresář k vyzvednutí tohoto souboru.

   Soubor amqsvmha.h obsahuje makra, která jsou použita kódem generovaným příkazem CRTMQCVX. Pokud struktura, která má být převedená, obsahuje znaková data, tato makra volají aplikaci MQXCNVC.
5. Najděte následující rámečky komentáře ve zdrojovém souboru a vložte kód podle popisu:

   a. Směrem ke konci zdrojového souboru, rámeček komentáře začíná znaky:

   ```
   /* Insert the functions produced by the data-conversion exit */
   ```

   Zde vložte fragment kódu vygenerovaný v kroku “3” na stránce 957.

b. V blízkosti středu zdrojového souboru začíná rámeček s komentářem:

```
/* Insert calls to the code fragments to convert the format's */
```

Po této akci bude následovat komentář k funkci ConverttagSTRUCT.

Změňte název funkce na název funkce, kterou jste přidali v kroku “5.a” na stránce 957. Chcete-li aktivovat funkci, odeberte znaky komentáře. Pokud existuje několik funkcí, vytvořte volání pro každou z nich.

c. V blízkosti začátku zdrojového souboru začíná rámeček s komentářem:

```
/* Insert the function prototypes for the functions produced by */
```

Zde vložte příkazy prototypu funkce pro funkce přidané v kroku “3” na stránce 957.

6. Vytvořte následující příkazový soubor:

```
cl -I  MQ_INSTALLATION_PATH\Tools\C\Include -Tp   \
MYFORMAT.C



MYFORMAT.DEF
```

kde *MQ_INSTALLATION_PATH* je adresář, kde je nainstalován produkt IBM MQ .

7. Vydejte příkazový soubor pro kompilaci uživatelské procedury jako soubor DLL.
8. Umístěte výstup do podadresáře exit pod datovým adresářem IBM MQ . Výchozí adresář pro instalaci vašich uživatelských procedur na 32bitových systémech je *MQ_DATA_PATH*\Exits a pro 64bitové systémy je *MQ_DATA_PATH*\Exits64

Cesta použitá k vyhledání uživatelských procedur pro převod dat je uvedena v registru. Složka registru je:

```
HKEY_LOCAL_MACHINE\SOFTWARE\IBM\WebSphere
MQ\Installation\MQ_INSTALLATION_NAME\Configuration\ClientExitPath\
```

a klíč registru je: ExitsDefaultPath. Tato cesta může být nastavena pro každého správce front a uživatelská procedura je vyhledána pouze v této cestě nebo v cestách.

Poznámka:

1. Pokud CRTMQCVX používá komprimované struktury, všechny aplikace IBM MQ musí být kompilovány tímto způsobem.
2. Ukončovací programy konverze dat musí být reentrantní.
3. MQXCNVC je jediné volání MQI, které lze vydat z uživatelské procedury pro převod dat.

Ukončit a přepnout soubory LOAD na operačních systémech Windows

Procesy správce front produktu IBM WebSphere MQ for Windows 7.5 jsou 32bitové. V důsledku toho při použití 64bitových aplikací musí mít některé typy výstupu a zaváděcí soubory XA k dispozici také 32bitovou verzi, kterou může správce front používat. Je-li 32bitová verze výstupního nebo zaváděcího souboru přepínače XA vyžadována a není k dispozici, dojde k selhání příslušného volání nebo příkazu rozhraní API.

Dva atributy jsou podporovány v qm.ini file pro *ExitPath*. Jedná se o ExitsDefaultPath= *MQ_INSTALLATION_PATH*\exits a ExitsDefaultPath64= *MQ_INSTALLATION_PATH*\exits64. *MQ_INSTALLATION_PATH* představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ . Pomocí těchto podmínek lze zajistit, že bude nalezena příslušná knihovna. Je-li uživatelská procedura použita v klastru IBM MQ , je také zajištěno, že bude nalezena odpovídající knihovna ve vzdáleném systému.

Následující tabulka obsahuje seznam různých typů souborů uživatelské procedury pro ukončení a přepínání a uvádí, zda jsou požadovány 32bitové nebo 64bitové verze, nebo obojí, podle toho, zda jsou používány 32bitové nebo 64bitové aplikace:

typy souboru	32bitové aplikace	64bitové aplikace
uživatelská procedura rozhraní API	32 bitů a 64 bitů	64bitová
Ukončení převodu dat	32bitové	64bitová
Uživatelské procedury kanálu serveru (všechny typy)	64bitová	64bitová
Uživatelské procedury kanálu klienta (všechny typy)	32bitové	64bitová
Ukončení instalovatelné služby	64bitová	64bitová
Ukončení modulu WLM klastru	64bitová	64bitová
Publikování/odběr-ukončení směrování	64bitová	64bitová
Soubory načtení přepínače databáze	32 bitů a 64 bitů	64bitová
Knihovny produktu External Transaction Manager AX	32bitové	64bitová
Uživatelská procedura před připojením	32bitové	64bitová

Odkazování na definice připojení pomocí předání před připojením z úložiště

Produkt IBM MQ MQI clients lze nakonfigurovat tak, aby vyhledal úložiště za účelem získání definic připojení s použitím knihovny uživatelské procedury před připojením.

Úvod

Klientská aplikace se může připojit ke správci front pomocí tabulek CCDT (Client Channel Definition tabulek). Obecně je soubor CCDT umístěn na centrálním síťovém souborovém serveru a na něm klienti odkazují. Vzhledem k tomu, že je obtížné spravovat a spravovat různé klientské aplikace odkazující na soubor CCDT, je flexibilním přístupem ukládat definice klientů do globálního úložiště, jako je adresář LDAP, WebSphere Registr a úložiště nebo jiné úložiště. Uložení definic připojení klienta v úložišti usnadňuje správu definic připojení klienta a aplikace mohou přistupovat ke správným a nejaktuálnějším definicím připojení klienta.

Během provádění volání MQCONN/X načte produkt IBM MQ MQI client určenou výstupní knihovnu před připojením a vyvolá uživatelskou funkci pro načtení definic připojení. Načtené definice připojení se pak použijí k navázání spojení se správcem front. Podrobnosti o knihovně uživatelské procedury a funkci k vyvolání jsou určeny v konfiguračním souboru mqclient.ini .

Syntaxe

`void MQ_PRECONNECT_EXIT` (*pExitParms*, *pQMgrName*, *ppConnectOpts*, *pCompCode*, *pReason*);

Parametry

Parametry příkazu pExit

Typ: vstup PMQNXP /výstup

Struktura konfiguračního parametru **`PreConnection`** .

Tato struktura je přidělována a udržována volajícím pro ukončení.

Název pQMgr
Typ: PMQCHAR vstupní/výstupní

Název správce front.

Na vstupu je tento parametr řetězec filtru dodávaný do volání rozhraní API MQCONN prostřednictvím parametru **QMgrName** . Toto pole může být prázdné, explicitní nebo obsahovat určité zástupné znaky. Pole je změněno uživatelskou procedurou. Při volání procedury MQXR_TERM je parametr nastaven na hodnotu Null.

ppConnectOpts
Typ: ppConnectOpts vstup/výstup

Volby, které řídí akci MQCONNX.

Jedná se o ukazatel na strukturu voleb připojení MQCNO, která řídí akci volání rozhraní API MQCONN. Při volání procedury MQXR_TERM je parametr nastaven na hodnotu Null. Klient MQI vždy poskytuje strukturu MQCNO pro ukončení i v případě, že ji aplikace původně neposkytovala. Pokud aplikace poskytuje strukturu MQCNO, klient vytvoří duplikát a předá jej do uživatelské procedury, kde je upravena. Klient si zachová vlastnictví objektu MQCNO.

MQCD, na které odkazuje objekt MQCNO, má přednost před jakoukoli definicí připojení poskytnutou prostřednictvím pole. Klient používá strukturu MQCNO k připojení ke správci front a ostatní jsou ignorovány.

Kód pComp
Typ: PMQLONG vstupní/výstupní

Kód dokončení.

Ukazatel na MQLONG, který přijímá kód dokončení ukončení. Musí se jednat o jednu z následujících hodnot:

- MQCC_OK -Úspěšné dokončení.
- MQCC_WARNING -Varování (částečné dokončení)
- MQCC_FAILED -Volání se nezdařilo.

pReason
Typ: PMQLONG vstupní/výstupní

Kód určující kvalifikaci pCompCode.

Ukazatel na hodnotu MQLONG, která přijímá kód příčiny ukončení. Je-li kód dokončení MQCC_OK, jediná platná hodnota je:

- MQRC_NONE-(0, x '000') Chybí důvod k vytvoření sestavy.

Je-li kód dokončení MQCC_FAILED nebo MQCC_WARNING, může funkce uživatelské procedury nastavit pole s kódem příčiny na jakoukoli platnou hodnotu MQRC_ *.

Vyvolání jazyka C

```
void MQ_PRECONNECT_EXIT (&ExitParms, &QMgrName, &pConnectOpts, &CompCode, &Reason);
```

Parameter

```
PMQNXP  pExitParms   /*PreConnect exit parameter structure*/
PMQCHAR pQMgrName    /*Name of the queue manager*/
PPMQCNO ppConnectOpts/*Options controlling the action of MQCONNX*/
PMQLONG pCompCode    /*Completion code*/
PMQLONG pReason      /*Reason qualifying pCompCode*/
```

Zápis a kompilace uživatelských procedur pro publikování

Chcete-li změnit obsah publikované zprávy před jejím přijetím odběrateli, můžete nakonfigurovat uživatelskou proceduru publikování ve správci front. Můžete také změnit záhlaví zprávy nebo nedoručit zprávu do odběru.

Poznámka: Uživatelské procedury publikování nejsou v produktu z/OSpodporovány.

Pomocí uživatelské procedury pro publikování můžete zkontrolovat a změnit zprávy doručené odběratelům:

- Prozkoumat obsah zprávy publikované pro každého odběratele
- Upravit obsah zprávy publikované pro každého odběratele
- Změnit frontu, do níž je zpráva vložena
- Zastavit doručení zprávy odběrateli

Psaní uživatelské procedury publikování

Kroky uvedené v části "Zapisování uživatelských procedur a instalovatelných služeb na UNIX, Linux a Windows" na stránce 898vám pomohou při zápisu a kompilaci vaší uživatelské procedury.

Poskytovatel uživatelské procedury pro publikování definuje, co bude uživatelská procedura dělat. Ukončení se však musí podřídit pravidlům definovaným v souboru MQPSXP.

Produkt IBM MQ neposkytuje implementaci vstupního bodu MQ_PUBLISHER_EXIT. Poskytuje deklaraci typedef jazyka C. Použijte typedef k deklarování parametrů pro uživatelskou proceduru správně. Následující příklad ukazuje, jak použít deklaraci typedef:

```
#include "cmqec.h"

MQ_PUBLISH_EXIT MyPublishExit;

void MQENTRY MyPublishExit( PMQPSXP pExitParms,
                            PMQPBC  pPubContext,
                            PMQSBC  pSubContext )
{
/* C language statements to perform the function of the exit */
}
```

Uživatelská procedura publikování se spustí v rámci procesu správce front jako výsledek následujících operací:

- Operace publikování, kde je zpráva doručena jednomu nebo více odběratelům.
- Odebírat operaci, kde je doručena jedna nebo více zachovaných zpráv
- Operace Požadavek na odběr, ve které je doručena jedna nebo více zachovaných zpráv

Je-li uživatelská procedura publikování volána pro připojení, je nastavena první hodnota, která se nazývá *ExitReason* , je nastavena na hodnotu MQXR_INIT . Než se připojení odpojí po použití uživatelské procedury publikování, je uživatelská procedura volána s kódem *ExitReason* MQXR_TERM.

Je-li uživatelská procedura publikování konfigurována, ale nelze ji načíst při spuštění správce front, jsou pro správce front zakázány operace publikování/odběru zpráv. Před povolením systému zpráv publikování/odběru je třeba opravit problém nebo restartovat správce front.

Každé připojení IBM MQ , které vyžaduje ukončení publikování, může selhat při načtení nebo inicializaci uživatelské procedury. Pokud se ukončení nepodaří načíst nebo inicializovat, operace publikování/odběru, které vyžadují uživatelskou proceduru publikování, jsou pro toto připojení zakázány. Operace se nezdaří s kódem příčiny IBM MQ MQRC_PUBLISH_EXIT_ERROR.

Kontext, ve kterém je volána uživatelská procedura publikování, je připojení aplikace ke správci front. Oblast uživatelských dat je spravována správcem front pro každé připojení, které provádí operace publikování. Uživatelská procedura může uchovávat informace v oblasti uživatelských dat pro každé připojení.

Uživatelská procedura publikování může používat některá volání MQI. Může používat pouze ta volání MQI, která manipulují s vlastnostmi zprávy. Volání jsou:

- MQBUFMH5
- MQCRTM
- MQDLTMH
- MQDLTMP
- MQMBUF
- MQINQMP
- MQSETMP

Pokud uživatelská procedura publikování změní cílového správce front nebo název fronty, neprovádí se žádná nová kontrola oprávnění.

Kompilování uživatelské procedury publikování

Uživatelská procedura publikování je dynamicky načtenou knihovnou; lze ji považovat za uživatelskou proceduru kanálu. Informace o kompilaci uživatelských procedur viz "Zapisování uživatelských procedur a instalovatelných služeb na UNIX, Linux a Windows" na stránce 898.

Ukázka uživatelské procedury publikování

Ukázkový ukončovací program se nazývá `amqspse0.c`. Do souboru protokolu bude zapsána jiná zpráva v závislosti na tom, zda byla uživatelská procedura volána pro operace inicializace, publikování nebo ukončení. Také demonstruje použití pole uživatelské oblasti pro ukončení k tomu, aby bylo možné vhodně přidělit a uvolnit paměť.

Konfigurace uživatelských procedur publikování

Chcete-li konfigurovat uživatelskou proceduru publikování, musíte definovat určité atributy.

V systémech Windows a Linux můžete k definování atributů použít Průzkumníka IBM MQ . Atributy jsou definovány na stránce vlastností správce front v části Publikování/odběr.

Chcete-li nakonfigurovat uživatelskou proceduru publikování v souboru `qm.ini` v systémech UNIX and Linux , vytvořte oddíl s názvem `PublishSubscribe`. Stanza `PublishSubscribe` má následující atributy:

PublishExitCesta = [cesta] |*název_modulu*
Název modulu a cesta obsahující kód uživatelské procedury publikování. Maximální délka tohoto pole je `MQ_EXIT_NAME_LENGTH`. Předvolba je žádná uživatelská procedura publikování.

PublishExitFunkce = *název_funkce*
Název vstupního bodu funkce do modulu, který obsahuje výstupní kód publikování. Maximální délka tohoto pole je `MQ_EXIT_NAME_LENGTH`.

IBM i Pokud se v systému IBM ipoužívá program, vynechte `PublishExitFunction`.

PublishExitData = *řetězec*
Pokud správce front volá uživatelskou proceduru pro publikování, předá strukturu `MQPSXP` jako vstup. Data zadaná pomocí atributu **PublishExitData** se poskytují v poli *ExitData* struktury. Řetězec může mít délku až `MQ_EXIT_DATA_LENGTH` znaků. Předvolba je 32 prázdných znaků.

Zápis a kompilace uživatelských procedur pracovní zátěže klastru

Chcete-li přizpůsobit správu pracovní zátěže klastrů, zapište výstupní program pracovní zátěže klastru. Při směrování zpráv můžete při směrování zpráv brát v úvahu náklady na použití kanálu v různých denních dobách nebo při směrování zprávy. Jedná se o faktory, které nejsou brány v úvahu pro standardní algoritmus správy pracovní zátěže.

Ve většině případů je algoritmus správy pracovní zátěže dostatečný pro vaše potřeby. Nicméně, abyste mohli poskytnout vlastní uživatelský program pro přizpůsobení správy pracovní zátěže, produkt IBM MQ zahrnuje uživatelskou proceduru a uživatelská procedura pracovní zátěže klastru.

Můžete mít některé konkrétní informace o své síti nebo zprávách, které byste mohli použít k ovlivnění vyrovnávání pracovní zátěže. Možná víte, které jsou vysokokapacitní kanály nebo levné přenosové cesty k síti, nebo můžete chtít směrovat zprávy v závislosti na jejich obsahu. Můžete se rozhodnout, zda chcete napsat uživatelský program pracovní zátěže klastru, nebo použít některý z nich dodaný třetí stranou.

Uživatelská procedura pracovní zátěže klastru je volána při přístupu ke frontě klastru. Nazývá se to MQOPEN, MQPUT1 a MQPUT.

Cílový správce front vybraný v době MQOPEN je pevný, je-li zadán MQOO_BIND_ON_OPEN . V tomto případě je uživatelská procedura spuštěna pouze jednou.

Pokud není správce cílové fronty opraven v době MQOPEN , je cílový správce front vybrán v době volání příkazu MQPUT . Pokud cílový správce front není k dispozici nebo selže, zatímco zpráva je stále v přenosové frontě, je uživatelská procedura volána znovu. Je vybrán nový cílový správce front. Pokud kanál zpráv selže během přenosu zprávy a je vrácena zpráva, je vybrán nový cílový správce front.

Multi V systému Multiplatformysprávce front načte novou uživatelskou proceduru pracovní zátěže klastru při příštím spuštění správce front.

Pokud definice správce front neobsahuje název ukončovacího programu pracovní zátěže klastru, nebude uživatelská procedura pracovní zátěže klastru volána.

Do ukončení pracovní zátěže klastru ve struktuře výstupního parametru MQWXPse předávají různé údaje:

- Struktura definice zprávy, MQMD.
- Parametr délky zprávy.
- Kopie zprávy nebo část zprávy.

Na platformách jiných nežz/OS , pokud používáte produkt CLWLMode=FAST, každý proces operačního systému načte vlastní kopii uživatelské procedury. Různá připojení ke správci front mohou způsobit vyvolání různých kopií uživatelské procedury. Je-li uživatelská procedura spuštěna ve výchozím bezpečném režimu CLWLMode=SAFE, je jedna kopie uživatelské procedury spuštěna ve svém vlastním samostatném procesu.

Zápis uživatelských procedur pracovní zátěže klastru

z/OS Informace o zápisu uživatelských procedur pracovní zátěže klastru pro produkt z/OSnaleznete v tématu “Programování uživatelské procedury pracovní zátěže klastru pro produkt IBM MQ for z/OS” na stránce 965.

Multi Pro více platforem nesmí uživatelská procedura pracovní zátěže klastru používat volání MQI. V jiných ohledech jsou pravidla pro zápis a kompilaci ukončovacích programů pracovní zátěže klastru podobná pravidlům, která se používají pro ukončovací programy kanálu. Postupujte podle kroků v části “Zapisování uživatelských procedur a instalovatelných služeb na UNIX, Linux a Windows” na stránce 898a použijte ukázkový program, “Ukázková uživatelská procedura pracovní zátěže klastru” na stránce 964 pro usnadnění zápisu a kompilaci uživatelské procedury.

Další informace o uživatelských procedurách kanálů naleznete v tématu “Psaní programů výstupních bodů kanálu” na stránce 929.

Konfigurace uživatelských procedur pracovní zátěže klastru

Pracovní zátěž klastru se ukončí v definici správce front zadáním atributu uživatelské procedury pracovní zátěže klastru u příkazu ALTER QMGR . Příklad:

```
ALTER QMGR CLWLEXIT(myexit)
```

Související informace

Volání uživatelské procedury pracovní zátěže klastru a datové struktury

Ukázková uživatelská procedura pracovní zátěže klastru

Produkt IBM MQ obsahuje ukázkový výstupní program pracovní zátěže klastru. Vzorek můžete zkopírovat a použít jako základ pro své vlastní programy.

z/OS **IBM MQ for z/OS**

Ukázkový uživatelský program pracovní zátěže klastru je dodáván v Assembleru a v produktu C. Verze Assembler se nazývá CSQ4BAF1 a lze ji nalézt v knihovně `thlqual.SCSQASMS`. Verze C se nazývá CSQ4BCF1 a lze ji nalézt v knihovně `thlqual.SCSQC37S`. `thlqual` je vysokoúrovňový kvalifikátor cílové knihovny pro datové sady produktu IBM MQ ve vaší instalaci.

Multi **IBM MQ for Multiplatforms**

Ukázkový uživatelský program pracovní zátěže klastru se dodává v C a nazývá se `amqswlm0.c`. Je možné jej nalézt v:

Tabulka 129. Ukázka umístění uživatelského programu pracovní zátěže klastru pro více platforem

Platforma	**Cesta k souboru**
AIX, HP-UX, Sun Solaris	*MQ_INSTALLATION_PATH*`/samp`
Windows	*MQ_INSTALLATION_PATH* `\Tools\c\Samples`
IBM i IBM i IBM i	Knihovna `qmqm`

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Tato ukázková uživatelská procedura směruje všechny zprávy do konkrétního správce front, pokud nebude tento správce front nedostupný. Reaguje na selhání správce front přesměrováním zpráv do jiného správce front.

Označit, do kterého správce front se mají zasílat zprávy. Zadejte název přijímacího kanálu klastru do atributu `CLWLDATA` v definici správce front. Příklad:

```
ALTER QMGR CLWLDATA(' my-cluster-name. my-queue-manager ')
```

Chcete-li tuto proceduru povolit, zadejte její úplnou cestu a název do atributu `CLWLEXIT` :

Linux UNIX V systému UNIX and Linux:

```
ALTER QMGR CLWLEXIT(' path /amqswlm(clwlFunction)')
```

Windows V systému Windows:

```
ALTER QMGR CLWLEXIT(' path \amqswlm(clwlFunction)')
```

z/OS V systému z/OS:

```
ALTER QMGR CLWLEXIT(CSQ4BxF1)
```

kde x je buď `'A'` , nebo `'C'`, v závislosti na programovacím jazyce verze, kterou používáte.

IBM i V systému IBM ipoužijte některý z následujících příkazů:

- Použijte příkaz MQSC:

```
ALTER QMGR CLWLEXIT('AMQSWLM       library       ')
```

  Jak název programu, tak název knihovny obsadí 10 znaků a jsou-li to nezbytné, jsou vyplněny mezerami.
- Použijte CL příkaz:

```
CHGMQM MQMNAME( qmgrname ) CLWLEXIT(' library /AMQSWLM')
```

Nyní místo použití dodaného algoritmu správy pracovní zátěže zavolá produkt IBM MQ tuto proceduru, aby přesměroval všechny zprávy do vybraného správce front.

Programování uživatelské procedury pracovní zátěže klastru pro produkt IBM MQ for z/OS

Uživatelské procedury pracovní zátěže klastru jsou při použití příkazu z/OS **LINK** vyvolány jako příkazy. Východy se řídí řadou přísných pravidel programování. Vyhněte se použití většiny příkazů SVC, které zahrnují čekání, nebo použití STAE nebo ESTAE v uživatelské proceduře pracovní zátěže.

Uživatelské procedury pracovní zátěže klastru jsou vyvolány jako by z/OS **LINK** v:

- Stav neautorizovaného problémového programu
- Řídicí režim primárního adresního prostoru
- Režim non-cross-memory
- Režim registru bez přístupu
- 31bitový režim adresování
- Paměťový klíč 8
- Maska klíče programu 8
- Klíč TCB 8

Vložte moduly s upravenými odkazy do datové sady určené příkazem CSQXLIB DD procedury adresního prostoru správce front. Názvy zaváděcích modulů jsou určeny jako názvy procedur pracovní zátěže v definici správce front.

Při zápisu uživatelských procedur pracovní zátěže pro produkt IBM MQ for z/OSplatí následující pravidla:

- Musíte zapsat uživatelské procedury v assembleru nebo C. Pokud používáte jazyk C, musí odpovídat programovacím prostředím systému C pro ukončení systému, popsané v příručce *z/OS C/C++ Programming Guide*, SC09-4765.
- Pokud používáte volání MQXCLWLN , propojte úpravy s CSQMFCLWdodanými v *thlqual*.SCSQLOAD.
- Uživatelské procedury jsou načteny z neautorizovaných knihoven definovaných pomocí příkazu CSQXLIB DD . Zadání CSQXLIB má DISP=SHR, ukončení může být aktualizováno, zatímco je spuštěn správce front, s novou verzí použitou v dalším podprocesu produktu MQCONN , který správce front spouští.
- Východy musí být reentrantní a mohou být spuštěny kdekoli ve virtuálním úložišti.
- Uživatelské procedury musí resetovat prostředí při návratu na položku při vstupu.
- Opustí musí uvolnit jakékoli získané úložiště, nebo se ujistěte, že úložiště je uvolněno následným vyvoláním ukončení.
- Nejsou povolena žádná volání MQI.
- Ukončení nesmí používat žádné systémové služby, které by mohly způsobit čekání, protože čekání vážně degraduje výkonnost správce front. Obecně řečeno, vyhněte se SVC, PC nebo I/O.
- Výstupy nesmí vydávat ESTAE nebo SPIE, kromě jakýchkoli podúloh, které připojují.

Poznámka: Neexistují žádná absolutní omezení pro to, co lze provést při ukončení. Většina SVC však zahrnuje čekání, takže se jim vyhýbejte, s výjimkou následujících příkazů:

- **GETMAIN** / **FREEMAIN**
- **LOAD** / **DELETE**

Nepoužívejte ESTAE a ESPIEs, protože jejich ošetření chyb může kolidovat s ošetřením chyb prováděnými produktem IBM MQ. Produkt IBM MQ nemusí být schopen provést zotavení z chyby, nebo váš uživatelský program nemusí přijímat všechny informace o chybě.

Parametr systému EXITLIM omezuje dobu, po kterou může být ukončena uživatelská procedura. Výchozí hodnota pro EXITLIM je 30 sekund. Pokud se zobrazí návratový kód MQRC_CLUSTER_EXIT_ERROR, 2266 X'8DA' , může dojít k zacyklení vaší uživatelské procedury. Pokud si myslíte, že ukončení potřebuje dokončení více než 30 sekund, zvyšte hodnotu EXITLIM.

Sestavení procedurální aplikace

Aplikaci IBM MQ můžete psát v jednom z několika procedurálních jazyků a aplikaci spustit na několika různých platformách.

AIX Sestavuje se vaše procedurální aplikace na AIX

Publikace AIX popisují, jak vytvářet spustitelné aplikace z programů, které napíšete.

Toto téma popisuje další úlohy a změny standardních úloh, které musíte provést při sestavování aplikací produktu IBM MQ for AIX , které mají být spuštěny v rámci produktu AIX. Volby C, C++ a COBOL jsou podporovány. Informace o přípravě programů C + +, viz Použití C++.

Úlohy, které musíte provést, abyste vytvořili spustitelnou aplikaci pomocí produktu IBM MQ for AIX , se liší pomocí programovacího jazyka, ve kterém je váš zdrojový kód zapsán. Kromě kódování volání MQI ve vašem zdrojovém kódu musíte přidat příslušné jazykové příkazy, aby bylo možné zahrnout soubory začlenění produktu IBM MQ for AIX do jazyka, který používáte. Seznamte se s obsahem těchto souborů. Úplný popis najdete v tématu “Soubory definic dat produktu IBM MQ” na stránce 686 .

Při spuštění serveru se závitem nebo klientskými aplikacemi s podporou podprocesů nastavte proměnnou prostředí AIXTHREAD_SCOPE = S.

Příprava programů jazyka C v produktu AIX

Toto téma obsahuje informace o propojování knihoven nezbytných pro přípravu programů v jazyce C v systému AIX.

Předkompilované C programy jsou dodávány v adresáři *MQ_INSTALLATION_PATH*/samp/bin . Použijte kompilátor ANSI a spusťte následující příkazy. Další informace o programování 64bitových aplikací najdete v tématu Kodo standardy na 64bitových platformách.

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Pro 32 bitové aplikace:

```
$ xlc_r -o amqsput_32 amqsput0.c -I MQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -lmqm
```

kde amqsput0 je ukázkový program.

Pro 64bitové aplikace:

```
$ xlc_r -q64 -o amqsput_64 amqsput0.c -I MQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib64
-lmqm
```

kde amqsput0 je ukázkový program.

Pokud používáte kompilátor VisualAge C/C++ pro programy C + +, musíte zahrnout volbu -q namemangling=v5 , abyste získali všechny symboly IBM MQ vyřešené při propojování knihoven.

Chcete-li používat programy na počítači, který má nainstalované pouze IBM MQ MQI client for AIX , znovu zkompilujte programy tak, aby je propojíte s knihovnou klienta (-lmqic).

Propojování knihoven

Potřebujete následující knihovny:

- Propojte své programy s příslušnou knihovnou poskytnutou produktem IBM MQ.

  V prostředí bez podprocesů se připojte k jedné z následujících knihoven:

Soubor knihovny	Typ programu/ukončení
libmqm.a	Server pro C
libmqic.a & libmqm.a	Klient pro C

  V prostředí s podprocesy se připojte k jedné z následujících knihoven:

Soubor knihovny	Typ programu/ukončení
libmqm_r.a	Server pro C
libmqic_r.a & libmqm_r.a	Klient pro C

  Chcete-li například sestavit jednoduchou aplikaci IBM MQ z jedné kompilační jednotky, spusťte následující příkazy.

  Pro 32 bitové aplikace:

```
$ xlc_r -o amqsputc_32_r amqsput0.c -I MQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -lmqm_r
```

  kde `amqsput0` je ukázkový program.

  Pro 64bitové aplikace:

```
$ xlc_r -q64 -o amqsputc_64_r amqsput0.c -I MQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib64 -lmqm_r
```

  kde `amqsput0` je ukázkový program.

  Chcete-li používat programy na počítači, který má nainstalované pouze IBM MQ MQI client for AIX , znovu zkompilujte programy tak, aby je propojíte s knihovnou klienta (`-lmqic`).

  Poznámka:

  1. Nemůžete odkazovat na více než jednu knihovnu. To znamená, že nemůžete současně propojit jak se závity, tak i bez podprocesové knihovny.
  2. Pokud píšete instalovatelnou službu (viz Správa), musíte se odkázat na knihovnu `libmqmzf.a` v aplikaci bez podprocesů a do knihovny `libmqmzf_r.a` v aplikaci s podprocesy.
  3. Pokud vytváříte aplikaci pro externí koordinaci správce transakcí kompatibilní se standardem XA, jako např. IBM TXSeries, Encina nebo BEA Tuxedo, musíte se připojit k serveru `libmqmxa.a` (nebo `libmqmxa64.a` , pokud váš správce transakcí zachází s typem " long ' typu 64 bit) a `libmqz.a` v aplikaci bez podprocesů a do `libmqmxa_r.a` (nebo `libmqmxa64_r.a`) a `libmqz_r.a` knihoven v aplikaci se závitem.
  4. Musíte propojit důvěryhodné aplikace se závitem IBM MQ knihoven. Nicméně v jednom okamžiku může být připojeno pouze jedno vlákno v důvěryhodné aplikaci na systému IBM MQ v systému UNIX and Linux .
  5. Musíte propojit knihovny produktu IBM MQ před všemi ostatními knihovnami produktu.

Příprava programů COBOL v produktu AIX

Tyto informace použijte při přípravě programů v jazyce COBOL v produktu AIX pomocí sady IBM COBOL Set a Micro Focus COBOL.

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

- 32bitové příručky pro kopírování v COBOLu jsou instalovány v následujícím adresáři:

```
MQ_INSTALLATION_PATH/inc/cobcpy32
```

  a symbolické odkazy jsou vytvořeny v:

```
MQ_INSTALLATION_PATH/inc
```

- 64bitové příručky pro kopírování v jazyce COBOL jsou instalovány v následujícím adresáři:

```
MQ_INSTALLATION_PATH/inc/cobcpy64
```

V následujících příkladech nastavte proměnnou prostředí **COBCPY** na:

```
MQ_INSTALLATION_PATH/inc/cobcpy32
```

pro 32bitové aplikace a:

```
MQ_INSTALLATION_PATH/inc/cobcpy64
```

pro 64bitové aplikace.

Musíte propojit svůj program s jedním z následujících souborů knihovny:

Soubor knihovny	Typ programu/ukončení
libmqmcb.a	Server pro COBOL (aplikace bez podprocesů)
libmqmcb_r.a	Server pro COBOL (podprocesová aplikace)
libmqicb.a	Klient pro COBOL (aplikace bez podprocesů
libmqicb_r.a	Klient pro COBOL (podprocesová aplikace)

V závislosti na programu můžete použít kompilátor IBM COBOL Set nebo Micro Focus COBOL compiler.

- Programy začínající `amqm` jsou vhodné pro kompilátor Micro Focus COBOL a
- Programy začínající `amq0` jsou vhodné pro kompilátor.

Příprava programů v jazyce COBOL pomocí sady IBM COBOL Set pro AIX

Ukázkové programy v jazyku COBOL jsou dodávány s produktem IBM MQ. Chcete-li zkompilovat takový program, zadejte příslušný příkaz z následujícího seznamu:

32bitová serverová aplikace bez podprocesů

```
$ cob2 -o amq0put0 amq0put0.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb -qLIB \
-ICOBCPY_VALUE
```

32bitová klientská aplikace bez podprocesů

```
$ cob2 -o amq0put0 amq0put0.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb -qLIB \
-ICOBCPY_VALUE
```

32bitová serverová serverová aplikace

```
$ cob2_r -o amq0put0 amq0put0.cbl -qTHREAD -L MQ_INSTALLATION_PATH/lib \
-lmqmcb_r -qLIB -ICOBCPY_VALUE
```

Aplikace klienta s 32bitovým podprocesem

```
$ cob2_r -o amq0put0 amq0put0.cbl -qTHREAD -L MQ_INSTALLATION_PATH/lib \
-lmqicb_r -qLIB -ICOBCPY_VALUE
```

64bitová serverová aplikace bez podprocesů

```
$ cob2 -o amq0put0 amq0put0.cbl -q64 -L MQ_INSTALLATION_PATH/lib - lmqmcb \
-qLIB -ICOBCPY_VALUE
```

Aplikace klienta s 64bitovým nevláknem

```
$ cob2 -o amq0put0 amq0put0.cbl -q64 -L MQ_INSTALLATION_PATH/lib - lmqicb \
-qLIB -ICOBCPY_VALUE
```

64bitová serverová serverová aplikace

```
$ cob2_r -o amq0put0 amq0put0.cbl -q64 -qTHREAD -L MQ_INSTALLATION_PATH/lib \
-lmqmcb_r -qLIB -ICOBCPY_VALUE
```

Aplikace klienta s 64bitovým vláknem

```
$ cob2_r -o amq0put0 amq0put0.cbl -q64 -qTHREAD -L MQ_INSTALLATION_PATH/lib \
-lmqicb_r -qLIB -ICOBCPY_VALUE
```

Příprava programů v jazyce COBOL pomocí Micro Focus COBOL

Před kompilací programu nastavte proměnné prostředí následujícím způsobem:

```
export COBCPY=COBCPY_VALUE
export LIBPATH=MQ_INSTALLATION_PATH/lib:$LIBPATH
```

Chcete-li zkompilovat 32bitového programu COBOL pomocí Micro Focus COBOL, zadejte:

- Server pro COBOL

```
$ cob32 -xvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb
```

- Klient pro COBOL

```
$ cob32 -xvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb
```

- Threadovaný server pro COBOL

```
$ cob32 -xtvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb_r
```

- Klient s podporou podprocesů pro COBOL

```
$ cob32 -xtvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb_r
```

Chcete-li zkompilovat 64bitový program COBOL pomocí Micro Focus COBOL, zadejte:

- Server pro COBOL

```
$ cob64 -xvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqmcb
```

- Klient pro COBOL

```
$ cob64 -xvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqicb
```

- Threadovaný server pro COBOL

```
$ cob64 -xtvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqmcb_r
```

- Klient s podporou podprocesů pro COBOL

```
$ cob64 -xtvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqicb_r
```

kde `amqminqx` je ukázkový program.

Popis proměnných prostředí, které je třeba nastavit, najdete v dokumentaci Micro Focus COBOL.

Příprava aplikačních programů produktu CICS v produktu AIX

Tyto informace použijte při přípravě programů produktu CICS v produktu AIX.

Použijte *přepínač XA* k propojení CICS s IBM MQ. Další informace o struktuře přepínačů XA najdete v tématu Struktury přepínačů XA.

K dispozici je ukázkový zdrojový kód zdrojového kódu, který umožňuje vývoj přepínačů XA pro další zprávy transakcí. Název poskytovaného modulu načtení přepínače je uveden v seznamuTabulka 130 na stránce 970.

Tabulka 130. Základní kód pro aplikační programy produktu CICS v systému AIX: inicializační rutina XA

Popis	**C (zdroj)**	**C (exec)-přidání do XAD.Stanza**
inicializační rutina XA	amqzscix.c	*amqzsc* - CICS pro AIX

Použijte předem sestavenou verzi zaváděcího souboru přepínače IBM MQ *amqzsc*, který se dodává spolu s produktem.

Vždy propojte transakce jazyka C se zabezpečenými podprocesy IBM MQ knihovny *libmqm_r.a.*, a vaše transakce v jazyce COBOL s knihovnou COBOL *libmqmcb_r.a.*.

You can find more information about supporting CICS transactions in the Správa IBM MQ System Administration Guide.

Podpora produktu TXSeriesCICS

Produkt IBM MQ v systému AIX podporuje prostředí TXSeries CICS pomocí rozhraní XA. Ujistěte se, že aplikace CICS jsou připojeny ke vláknové verzi knihoven IBM MQ .

Programy produktu CICS můžete spustit pomocí sady IBM COBOL Set for AIX nebo Micro Focus COBOL. Následující oddíly popisují rozdíl mezi spuštěnými programy produktu CICS na sadě COBOL IBM pro AIX a Micro Focus COBOL.

Zapište IBM MQ programy, které jsou zavedeny do stejné oblasti CICS buď v jazyce C, nebo v COBOLu. Nemůžete vytvořit kombinaci volání C a COBOL MQI do stejné oblasti CICS . Většina volání MQI ve druhém použitém jazyce se nezdařila s kódem příčiny `MQRC_HOBJ_ERROR`.

Příprava programů CICS COBOL pomocí sady IBM COBOL pro AIX

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Chcete-li použít produkt IBM COBOL, proveďte následující kroky:

1. Exportovat následující proměnnou prostředí:

```
export LDFLAGS="-qLIB -bI:/usr/lpp/cics/lib/cicsprIBMCOB.exp \
                -I MQ_INSTALLATION_PATH/inc -I/usr/lpp/cics/include \
                -e _iwz_cobol_main  \
```

kde `LIB` je direktiva kompilátoru.

2. Přeložte, zkompilujte a propojte program zadáním příkazu:

```
cicstcl -l IBMCOB yourprog.ccp
```

Příprava programů jazyka COBOL pro produkt CICS pomocí Micro Focus COBOL

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Chcete-li použít funkci Micro Focus COBOL, postupujte takto:

1. Přidejte modul knihovny běhové komponenty produktu IBM MQ COBOL do knihovny běhového prostředí pomocí následujícího příkazu:

```
cicsmkcobol -L/usr/lib/dce -L MQ_INSTALLATION_PATH/lib \
            MQ_INSTALLATION_PATH/lib/libmqmcbrt.o -lmqe_r
```

Poznámka: With `cicsmkcobol`, IBM MQ does not allow you to make MQI calls in the C programming language from your COBOL application.

Pokud mají existující aplikace taková volání, doporučuje se tyto funkce přesunout z aplikací v jazyce COBOL do své vlastní knihovny, například `myMQ.so`. Po přesunu funkcí nezahrnujte knihovnu IBM MQ `libmqmcbrt.o` při sestavování aplikace COBOL pro CICS.

Navíc, pokud vaše aplikace v jazyku COBOL nevolá žádné volání COBOL MQI, nepropojujte `libmqmz_r` s `cicsmkcobol`.

Tím se vytvoří soubor metod jazyka COBOL Micro Focus a povolí knihovně jazyka COBOL pro běhové prostředí produktu CICS volání IBM MQ v systémech UNIX and Linux .

Poznámka: Spusťte produkt `cicsmkcobol` pouze při instalaci jednoho z následujících produktů:

- Nová verze nebo vydání Micro Focus COBOL
- Nová verze nebo vydání CICS pro AIX
- Nová verze nebo vydání všech podporovaných databázových produktů (pouze pro transakce v jazyce COBOL)
- Nová verze nebo vydání IBM MQ

2. Exportovat následující proměnnou prostředí:

```
COBCPY= MQ_INSTALLATION_PATH/inc export COBCPY
```

3. Přeložte, zkompilujte a propojte program zadáním příkazu:

```
cicstcl -l COBOL -e yourprog.ccp
```

Příprava programů v programu CICS C

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Sestavte programy CICS C pomocí standardních zařízení produktu CICS :

1. Exportujte ***jeden*** z následujících proměnných prostředí:

- LDFLAGS = "-L/ *MQ_INSTALLATION_PATH* lib -lmqm_r" export LDFLAGS
- USERLIB = "-L *MQ_INSTALLATION_PATH* lib -lmqm_r" export USERLIB

2. Přeložte, zkompilujte a propojte program zadáním příkazu:

```
cicstcl -l C amqscic0.ccs
```

CICS vzorová transakce C

Ukázka zdroje C pro transakci produktu AIX IBM MQ je k dispozici v rámci AMQSCIC0.CCS. Transakce čte zprávy z přenosové fronty SYSTEM.SAMPLE.CICS.WORKQUEUE na výchozím správci front a umisťuje je do lokální fronty s názvem fronty, který je obsažen v záhlaví přenosu zprávy. Veškerá selhání se odesílají do fronty SYSTEM.SAMPLE.CICS.DLQ. Použijte ukázkový skript MQSC AMQSCIC0.TST pro vytvoření těchto front a ukázkových vstupních front.

Sestavuje se vaše procedurální aplikace na HP-UX

Tyto informace popisují další úlohy a změny standardních úloh, které musíte provést při sestavování aplikací produktu IBM MQ for HP-UX pro spouštění v produktu HP-UX.

Volby C, C++ a COBOL jsou podporovány. Informace o přípravě programů C + +, viz Použití C++.

Úlohy, které musíte provést, abyste vytvořili spustitelnou aplikaci pomocí produktu IBM MQ for HP-UX , se liší pomocí programovacího jazyka, ve kterém je váš zdrojový kód zapsán. Kromě kódování volání MQI ve vašem zdrojovém kódu musíte přidat příslušné jazykové příkazy, aby bylo možné zahrnout soubory začlenění produktu IBM MQ for HP-UX do jazyka, který používáte. Seznamte se s obsahem těchto souborů. Úplný popis najdete v tématu “Soubory definic dat produktu IBM MQ” na stránce 686 .

V celém tomto tématu používáme znak zpětného lomítka (\) k rozdělení dlouhých příkazů na více než jeden řádek. Nezadávejte tento znak; zadejte každý příkaz jako jednu řádku.

Příprava programů jazyka C v produktu HP-UX

Toto téma obsahuje informace, které je třeba zvážit při přípravě programů v jazyce C v produktu HP-UX; s příklady pro platformu IA64 (IPF).

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Pracujte ve svém normálním prostředí. Předkompilované C programy jsou dodávány v adresáři *MQ_INSTALLATION_PATH*/samp/bin .

Další informace o programování 64bitových aplikací najdete v tématu Kódování standardů na 64 bitových platformách.

Chcete-li používat TLS, musí být produkt IBM MQ MQI clients na systému HP-UX sestaven pomocí podprocesů POSIX .

Několik příkladů, které je třeba zvážit:

- “Platforma IA64 (IPF)” na stránce 972
- “Propojování knihoven” na stránce 974

Platforma IA64 (IPF)

Příklady sestavení amqsput0, cliexit a srvexit na platformě IA64(IPF).

Následující příklad sestaví ukázkový program amqsput0 jako klientskou aplikaci v 32bitovém prostředí bez podprocesů:

```
c89 -Wl,+b,: +e -D_HPUX_SOURCE -o amqsputc_32 amqsput0.c -I MQ_INSTALLATION_PATH/inc
  -L MQ_INSTALLATION_PATH/lib -L/usr/lib/hpux32 -lmqic
```

Následující příklad sestaví ukázkový program amqsput0 jako klientskou aplikaci v 32bitovém prostředí s podporou podprocesů:

```
c89 -mt -Wl,+b,: +e -D_HPUX_SOURCE -o amqsputc_32_r amqsput0.c -I MQ_INSTALLATION_PATH/inc
  -L MQ_INSTALLATION_PATH/lib -L/usr/lib/hpux32 -lmqic_r -lpthread
```

Následující příklad sestaví ukázkový program amqsput0 jako klientskou aplikaci v 64bitovém prostředí 64bitového prostředí:

```
c89 +DD64 +e -D_HPUX_SOURCE -o amqsputc_64 amqsput0.c -I MQ_INSTALLATION_PATH/inc
  -L MQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -lmqic
```

Následující příklad sestaví ukázkový program amqsput0 jako klientskou aplikaci v 64bitovém prostředí s podporou podprocesů:

```
c89 -mt +DD64 +e -D_HPUX_SOURCE -o amqsputc_64_r amqsput0.c -I MQ_INSTALLATION_PATH/inc
  -L MQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -lmqic_r -lpthread
```

Následující příklad sestaví ukázkový program amqsput0 jako serverovou aplikaci v 32bitovém prostředí bez podprocesů:

```
c89 -Wl,+b,: +e -D_HPUX_SOURCE -o amqsput_32 amqsput0.c -I MQ_INSTALLATION_PATH/inc
  -L MQ_INSTALLATION_PATH/lib -L/usr/lib/hpux32 -lmqm
```

Následující příklad sestaví ukázkový program amqsput0 jako serverovou aplikaci v 32bitovém prostředí s podporou podprocesů:

```
c89 -mt -Wl,+b,: +e -D_HPUX_SOURCE -o amqsput_32_r amqsput0.c -I MQ_INSTALLATION_PATH/inc
  -L MQ_INSTALLATION_PATH/lib -L/usr/lib/hpux32 -lmqm_r -lpthread
```

Následující příklad sestaví ukázkový program amqsput0 jako serverovou aplikaci v 64bitovém prostředí bez podprocesů.

```
c89 +DD64 +e -D_HPUX_SOURCE -o amqsput_64 amqsput0.c -I MQ_INSTALLATION_PATH/inc
  -L MQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -lmqm
```

Následující příklad sestaví ukázkový program amqsput0 jako serverovou aplikaci v prostředí s 64bitovým 64bitovým vláknem:

```
c89 -mt +DD64 +e -D_HPUX_SOURCE -o amqsput_64_r amqsput0.c -I MQ_INSTALLATION_PATH/inc
  -L MQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -lmqm_r -lpthread
```

V následujícím příkladu je založena uživatelská procedura ukončení klienta v 32bitovém prostředí bez podprocesů:

```
c89 +e +z -c -D_HPUX_SOURCE -o cliexit.o cliexit.c -I MQ_INSTALLATION_PATH/inc
ld +b: -b cliexit.o +ee MQStart -o /var/mqm/exits/cliexit_32 -L MQ_INSTALLATION_PATH/lib \
-L/usr/lib/hpux32 -lmqic
```

V následujícím příkladu je založena uživatelská procedura ukončení klienta v 32bitovém prostředí s podporou podprocesů:

```
c89 -mt +e +z -c -D_HPUX_SOURCE -o cliexit.o cliexit.c -I MQ_INSTALLATION_PATH/inc
ld +b: -b cliexit.o +ee MQStart -o /var/mqm/exits/cliexit_32_r -L MQ_INSTALLATION_PATH/lib \
-L/usr/lib/hpux32 -lmqic_r -lpthread
```

Následující příklad sestaví uživatelskou proceduru pro ukončení klienta v 64bitovém prostředí bez podprocesů:

```
c89 +DD64 +e +z -c -D_HPUX_SOURCE -o cliexit.o cliexit.c -I MQ_INSTALLATION_PATH/inc
ld -b cliexit.o +ee MQStart -o /var/mqm/exits64/cliexit_64 \
-L MQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -lmqic
```

Následující příklad sestaví uživatelskou proceduru pro ukončení klienta v 64bitovém prostředí se závitovým prostředím:

```
c89 -mt +DD64 +e +z -c -D_HPUX_SOURCE -o cliexit.o cliexit.c -I MQ_INSTALLATION_PATH/inc
ld -b cliexit.o +ee MQStart -o /var/mqm/exits/cliexit_64_r \
-L MQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -lmqic_r -lpthread
```

Následující příklad sestaví proceduru srvexit serveru v 32bitovém prostředí bez podprocesů se závity:

```
c89 +e +z -c -D_HPUX_SOURCE -o srvexit.o srvexit.c -I MQ_INSTALLATION_PATH/inc
ld +b: -b srvexit.o +ee MQStart -o /var/mqm/exits/srvexit_32 -L MQ_INSTALLATION_PATH/lib \
-L/usr/lib/hpux32 -lmqm
```

Následující příklad sestaví server srvexit serveru v 32bitovém prostředí s podporou podprocesů:

```
c89 -mt +e +z -c -D_HPUX_SOURCE -o srvexit.o srvexit.c -I MQ_INSTALLATION_PATH/inc
ld +b: -b srvexit.o +ee MQStart -o /var/mqm/exits/srvexit_32_r -L MQ_INSTALLATION_PATH/lib \
-L/usr/lib/hpux32 -lmqm_r -lpthread
```

Následující příklad sestaví proceduru srvexit serveru v 64bitovém prostředí 64bitového prostředí:

```
c89 +DD64 +e +z -c -D_HPUX_SOURCE -o srvexit.o srvexit.c -I MQ_INSTALLATION_PATH
MQ_INSTALLATION_PATH/inc
ld -b srvexit.o +ee MQStart -o /var/mqm/exits64/srvexit_64 \
-L MQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -lmqm
```

Následující příklad sestaví server srvexit serveru v 64bitovém prostředí s podporou podprocesů:

```
c89 -mt +DD64 +e +z -c -D_HPUX_SOURCE -o srvexit.o srvexit.c -I MQ_INSTALLATION_PATH/inc
ld -b srvexit.o +ee MQStart -o /var/mqm/exits/srvexit_64_r \
-L MQ_INSTALLATION_PATH/lib64 -L/usr/lib/hpux64 -lmqm_r -lpthread
```

Propojování knihoven

Je třeba propojit své programy s jednou z knihoven poskytnutých produktem IBM MQ.

Následující tabulka ukazuje, která knihovna se má použít v různých prostředích:

Hardwarová platforma	Prostředí s podprocesy nebo bez podprocesů	Typ programu/ukončení	Soubor knihovny
IA64 (IPF)	Vláknové	Server & Klient pro C	libmqm_r.so
IA64 (IPF)	Vláknové	Klient pro C	libmqic_r.so
IA64 (IPF)	Nevláknová	Server & Klient pro C	libmqm.so
IA64 (IPF)	Nevláknová	Klient pro C	libmqic.so

Poznámka:

1. Nemůžete odkazovat na více než jednu knihovnu. To znamená, že nemůžete současně propojit jak se závity, tak i bez podprocesové knihovny.
2. Pokud zapisujete instalovatelnou službu (další informace naleznete v příručce Správa), musíte se připojit ke knihovně produktu `libmqmzf.sl` .
3. Pokud vytváříte aplikaci pro externí koordinaci správce transakcí kompatibilní se standardem XA, jako např. IBM TXSeries Encina nebo BEA Tuxedo, musíte se připojit k serveru `libmqmxa.sl` (nebo

libmqmxa64.sl , pokud váš správce transakcí zachází s typem " long ' typu 64 bitů) a libmqz.sl v aplikaci bez podprocesů a do libmqmxa_r.sl (nebo libmqmxa64_r.sl). a libmqz_r.sl knihoven v aplikaci se závitem.

4. Musíte propojit knihovny produktu IBM MQ před všemi ostatními knihovnami produktu.

Příprava programů COBOL v produktu HP-UX

Informace o přípravě programů v jazyce COBOL v produktu HP-UXpomocí produktu Micro Focus Server Express s produktem IBM MQ na platformě IA64 (IPF) a spouštění programů v prostředí produktu IBM MQ MQI client .

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Poznámky k použití:

1. 32bitové příručky pro kopírování v COBOLu jsou instalovány v následujícím adresáři:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy32
   ```

   a symbolické odkazy jsou vytvořeny v:

   ```
   MQ_INSTALLATION_PATH/inc
   ```

2. 64bitové příručky pro kopírování v jazyce COBOL jsou instalovány v následujícím adresáři:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy64
   ```

3. V následujících příkladech nastavte COBCPY na:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy32
   ```

   pro 32bitové aplikace a:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy64
   ```

   pro 64bitové aplikace.

Proveďte kompilaci programů pomocí kompilátoru Micro Focus. Soubory kopií, které deklarují struktury, jsou v adresáři *MQ_INSTALLATION_PATH*/inc:

```
$ export LIB= MQ_INSTALLATION_PATH/lib:$LIB
$ export COBCPY="COBCPY_VALUE"
```

Kompilace 32bitových programů:

```
$ cob32 -xv amqsput.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb Server for COBOL
$ cob32 -xv amqsput.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb Client for COBOL
$ cob32 -xtv amqsput.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb_r Threaded Server for COBOL
$ cob32 -xtv amqsput.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb_r Threaded Client for COBOL
```

Kompilace 64bitových programů:

```
$ cob64 -xv amqsput.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqmcb Server for COBOL
$ cob64 -xv amqsput.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqicb Client for COBOL
$ cob64 -xtv amqsput.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqmcb_r Threaded Server for COBOL
$ cob64 -xtv amqsput.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqicb_r Threaded Client for COBOL
```

kde amqsput je ukázkový program.

Ujistěte se, že jste zadali odpovídající velikost zásobníku běhového prostředí; 16 kB je doporučené minimum.

Je třeba propojit své programy s příslušnou knihovnou poskytnutou produktem IBM MQ. Následující tabulka ukazuje, která knihovna se má použít v různých prostředích

Hardwarová platforma	Typ programu/ukončení	Soubor knihovny
IA64 (IPF)	Server pro COBOL	libmqmcb.so
IA64 (IPF)	Klient pro COBOL	libmqicb.so
IA64 (IPF)	Aplikace s podprocesy	libmqmcb_r.so

Použití produktu Micro Focus Server Express s produktem IBM MQ na platformě IA64 (IPF)

Podrobné informace o používání produktu Micro Focus Server Express ve spojení s produktem IBM MQ na platformě HP/IPF naleznete v příručce "Modely adresového prostoru podporované produktem IBM MQ for HP-UX na IA64 (IPF)" na stránce 977 .

Programy, které se mají spustit v prostředí produktu IBM MQ MQI client

Pokud používáte pro připojení klienta MQI k serveru logickou jednotku 6.2 , propojte svou aplikaci s produktem libsna.a, který je součástí produktu SNAplusAPI . Použijte volby `-lV3` a `-lstr` u příkazu pro kompilaci a propojení.

- Volba `-lV3` dává vašemu programu přístup k signální knihovně AT & T (SNAplusAPI používá AT & T signály)
- Volba `-lstr` propojí váš program s komponentou proudů.

Příprava programů produktu CICS v produktu HP-UX

Naučte se sestavovat transakční programy CICS v produktu HP-UX.

Chcete-li sestavit vzorovou transakci CICS , amqscic0.ccs, spusťte následující příkaz:

```
$ export USERLIB="-lmqm_r"
$ cicstcl -l C amqscic0.ccs
```

Je k dispozici modul přepínače XA, který vám umožňuje propojení CICS s IBM MQ:

Tabulka 131. Základní kód pro aplikace produktu CICS (HP-UX)

Popis	C (zdroj)	C (exec)
inicializační rutina XA	amqzscix.c	amqzsc

Další informace o podpoře CICS transakcí v příručce Správanajdete.

Podpora produktu TXSeriesCICS

Produkt IBM MQ v systému HP-UX podporuje prostředí TXSeries CICS pomocí rozhraní XA. Ujistěte se, že aplikace produktu CICS jsou připojeny ke vláknové verzi knihoven MQ .

Zapište IBM MQ programy, které jsou zavedeny do stejné oblasti CICS buď v jazyce C, nebo v COBOLu. Nemůžete vytvořit kombinaci volání C a COBOL MQI do stejné oblasti CICS . Většina volání MQI ve druhém použitém jazyce se nezdařila s kódem příčiny `MQRC_HOBJ_ERROR`.

CICS vzorová transakce C

Ukázka zdroje C pro transakci produktu CICS IBM MQ je k dispozici v rámci AMQSCIC0.CCS. Transakce čte zprávy z přenosové fronty SYSTEM.SAMPLE.CICS.WORKQUEUE na výchozím správci front a umisťuje je do lokální fronty s názvem fronty, který je obsažen v záhlaví přenosu zprávy. Veškerá selhání se odesílají do fronty SYSTEM.SAMPLE.CICS.DLQ. Použijte ukázkový skript MQSC AMQSCIC0.TST pro vytvoření těchto front a ukázkových vstupních front.

Příprava programů jazyka COBOL pro produkt CICS pomocí Micro Focus COBOL

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Chcete-li použít funkci Micro Focus COBOL, postupujte takto:

1. Přidejte modul knihovny běhové komponenty produktu IBM MQ COBOL do knihovny běhového prostředí pomocí následujícího příkazu:

   ```
   cicsmkcobol -L/usr/lib/dce -L MQ_INSTALLATION_PATH/lib \
               MQ_INSTALLATION_PATH/lib/libmqmcbrt.o -lmqe_r
   ```

   Poznámka: With `cicsmkcobol`, IBM MQ does not allow you to make MQI calls in the C programming language from your COBOL application.

   Pokud mají existující aplikace taková volání, doporučuje se tyto funkce přesunout z aplikací v jazyce COBOL do své vlastní knihovny, například `myMQ.so`. Po přesunu těchto funkcí nezahrnujte knihovnu IBM MQ `libmqmcbrt.o` při sestavování aplikace v jazyce COBOL pro produkt CICS.

   Navíc, pokud vaše aplikace v jazyku COBOL nevolá žádné volání COBOL MQI, nepropojujte `libmqmz_r` s `cicsmkcobol`.

   Tím se vytvoří soubor metod jazyka COBOL Micro Focus a povolí knihovně jazyka COBOL pro běhové prostředí produktu CICS volání IBM MQ v systémech UNIX and Linux .

   Poznámka: Spusťte produkt `cicsmkcobol` pouze při instalaci jednoho z následujících produktů:

   - Nová verze nebo vydání Micro Focus COBOL
   - Nová verze nebo vydání CICS pro HP-UX
   - Nová verze nebo vydání všech podporovaných databázových produktů (pouze pro transakce v jazyce COBOL)
   - Nová verze nebo vydání IBM MQ
2. Exportovat následující proměnnou prostředí:

   ```
   COBCPY= MQ_INSTALLATION_PATH/inc export COBCPY
   ```
3. Přeložte, zkompilujte a propojte program zadáním příkazu:

   ```
   cicstcl -l COBOL -e yourprog.ccp
   ```

Modely adresového prostoru podporované produktem IBM MQ for HP-UX na IA64 (IPF)

Produkt HP-UX nabízí několik modelů adresního prostoru, které mohou využívat aplikace produktu IBM MQ .

Produkt HP-UX podporuje dva modely adresního prostoru:

- MGAS-Převážně globální adresní prostor (jedná se o předvolbu a používá se pro IBM MQ)
- MPAS-převážně soukromý adresní prostor

Aplikace, které se připojují k produktu IBM MQ , mohou používat modely s adresní prostorem MGAS nebo MPAS. Aplikace vytvořené pomocí modelu MPAS, které se připojují k produktu IBM MQ s využitím sdílené

paměti, mohou způsobit menší náklady na výkon kvůli neefektivitě při mapování stránek sdílené paměti použité produktem IBM MQ do virtuálního adresního prostoru programu MPAS.

Aplikace v COBOLu sestavené pomocí standardně Micro Focus Server Express používají model MPAS jako model MPAS.

K ověření a změně modelu adresování používaného programem můžete použít program **chatr** .

Pokud se setkáte s problémy s připojením k produktu IBM MQ z 32bitového programu MPAS, zvažte použití modelu adresování MGAS nebo sestavení vaší aplikace jako 64-bitové aplikace MPAS, nikoli 32bitového aplikace MPAS.

Další podrobnosti o modelech s adresami MGAS a MPAS najdete v dokumentaci produktu HP-UX .

Linux Sestavuje se vaše procedurální aplikace na Linux

Tyto informace popisují další úlohy a změny standardních úloh, které musíte provést při sestavování aplikací produktu IBM MQ for Linux ke spuštění.

Jsou podporovány jazyky C a C++. Informace o přípravě programů C + +, viz Použití C++.

Příprava programů jazyka C v produktu Linux

Předkompilované C programy jsou dodávány v adresáři *MQ_INSTALLATION_PATH*/samp/bin . Chcete-li sestavit ukázku ze zdrojového kódu, použijte kompilátor gcc .

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Pracujte ve svém normálním prostředí. Další informace o programování 64bitových aplikací najdete v tématu Koding standardů na 64bitových platformách.

Propojování knihoven

V následujících tabulkách jsou uvedeny knihovny, které jsou zapotřebí při přípravě programů v jazyce C v systému Linux.

- Je třeba propojit své programy s příslušnou knihovnou poskytnutou produktem IBM MQ.

  V prostředí bez podprocesů se připojte pouze k jedné z následujících knihoven:

Soubor knihovny	Typ programu/ukončení
libmqm.so	Server pro C
libmqic.so & libmqm.so	Klient pro C

  V prostředí se závitem je odkaz pouze na jednu z následujících knihoven:

Soubor knihovny	Typ programu/ukončení
libmqm_r.so	Server pro C
libmqic_r.so & libmqm_r.so	Klient pro C

  Poznámka:

  1. Nemůžete odkazovat na více než jednu knihovnu. To znamená, že nemůžete současně propojit jak se závity, tak i bez podprocesové knihovny.
  2. Pokud zapisujete instalovatelnou službu (další informace naleznete v příručce Správa), musíte se připojit ke knihovně produktu libmqmzf.so .
  3. Pokud vytváříte aplikaci pro externí koordinaci správce transakcí kompatibilní se standardem XA, jako např. IBM TXSeries Encina nebo BEA Tuxedo, musíte se připojit k serveru libmqmxa.so (nebo libmqmxa64.so , pokud váš správce transakcí zachází s typem " long ' typu 64 bitů) a libmqz.so

v aplikaci bez podprocesů a do `libmqmxa_r.so` (nebo `libmqmxa64_r.so`). a `libmqz_r.so` knihoven v aplikaci se závitem.

4. Musíte propojit knihovny produktu IBM MQ před všemi ostatními knihovnami produktu.

Sestavení 31bitových aplikací
Toto téma obsahuje příklady příkazů používaných k sestavení 31bitových programů v různých prostředích.

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Klientská aplikace C, 31-bitů, bez podprocesů

```
gcc -m31 -o famqsputc_32 amqsput0.c -I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqic
```

Klientská aplikace C, 31bitový, s podporou podprocesů

```
gcc -m31 -o amqsputc_32_r amqsput0.c -I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqic_r -lpthread
```

Serverová aplikace C, 31bitový, bez podprocesů

```
gcc -m31 -o amqsput_32 amqsput0.c -I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqm
```

Serverová aplikace C, 31bitový, s podporou podprocesů

```
gcc -m31 -o amqsput_32_r amqsput0.c -I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqm_r -lpthread
```

Klientská aplikace C + +, 31bitový, bez podprocesů

```
g++ -m31 -fsigned-char -o imqsputc_32 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqc23gl
-limqb23gl -lmqic
```

klientská aplikace C + +, 31bitový, s podporou podprocesů

```
g++ -m31 -fsigned-char -o imqsputc_32_r imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqc23gl_r
-limqb23gl_r -lmqic_r -lpthread
```

Serverová aplikace C + +, 31 bitů, bez podprocesů.

```
g++ -m31 -fsigned-char -o imqsput_32 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqs23gl
-limqb23gl -lmqm
```

Serverová aplikace C + +, 31 bitů, s podprocesy

```
g++ -m31 -fsigned-char -o imqsput_32_r imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqs23gl_r
-limqb23gl_r -lmqm_r -lpthread
```

C client exit, 31-bit, non-threaded

```
gcc -m31 -shared -fPIC -o /var/mqm/exits/cliexit_32 cliexit.c
```

```
-I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib
-Wl,-rpath=/usr/lib -lmqic
```

C client exit, 31-bit, threaded

```
gcc -m31 -shared -fPIC -o /var/mqm/exits/cliexit_32_r cliexit.c
-I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib
-Wl,-rpath=/usr/lib -lmqic_r -lpthread
```

Ukončení serveru C, 31bitový, bez podprocesů.

```
gcc -m31 -shared -fPIC -o /var/mqm/exits/srvexit_32 srvexit.c
-I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib
-Wl,-rpath=/usr/lib -lmqm
```

Ukončení serveru C, 31bitový, s podporou podprocesů

```
gcc -m31 -shared -fPIC -o /var/mqm/exits/srvexit_32_r srvexit.c
-I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib
-Wl,-rpath=/usr/lib -lmqm_r -lpthread
```

Sestavování 32 bitových aplikací
Toto téma obsahuje příklady příkazů použitých pro sestavení 32bitových programů v různých prostředích.

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Klientská aplikace C, 32bitová, bez podprocesů

```
gcc -m32 -o amqsputc_32 amqsput0.c -I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqic
```

Klientská aplikace C, 32bitová, se závitem

```
gcc -m32 -o amqsputc_32_r amqsput0.c -I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqic_r -lpthread
```

Serverová aplikace C, 32bitová, bez podprocesů

```
gcc -m32 -o amqsput_32 amqsput0.c -I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqm
```

Serverová aplikace C, 32bitová, se závitem

```
gcc -m32 -o amqsput_32_r amqsput0.c -I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqm_r -lpthread
```

Klientská aplikace C + +, 32bitová, bez podprocesů

```
g++ -m32 -fsigned-char -o imqsputc_32 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqc23gl -limqb23gl -lmqic
```

Klientská aplikace C + +, 32bitová verze

```
g++ -m32 -fsigned-char -o imqsputc_32_r imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqc23gl_r -limqb23gl_r -lmqic_r -lpthread
```

Serverová aplikace C + +, 32bitová, bez podprocesů

```
g++ -m32 -fsigned-char -o imqsput_32 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqs23gl -limqb23gl -lmqm
```

Serverová aplikace C + +, 32bitová verze

```
g++ -m32 -fsigned-char -o imqsput_32_r imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqs23gl_r -limqb23gl_r -lmqm_r -lpthread
```

C client exit, 32bitový, non-threaded

```
gcc -m32 -shared -fPIC -o /var/mqm/exits/cliexit_32 cliexit.c
-I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib
-Wl,-rpath=/usr/lib -lmqic
```

C client exit, 32bitový, threaded

```
gcc -m32 -shared -fPIC -o /var/mqm/exits/cliexit_32_r cliexit.c
-I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib
-Wl,-rpath=/usr/lib -lmqic_r -lpthread
```

C server exit, 32bitový, bez podprocesů

```
gcc -m32 -shared -fPIC -o /var/mqm/exits/srvexit_32 srvexit.c -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib
-Wl,-rpath=/usr/lib -lmqm
```

C server exit, 32bitový, threaded

```
gcc -m32 -shared -fPIC -o /var/mqm/exits/srvexit_32_r srvexit.c
I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib
-Wl,-rpath=/usr/lib -lmqm_r -lpthread
```

Sestavování 64bitových aplikací
Toto téma obsahuje příklady příkazů používaných k sestavení 64bitových programů v různých prostředích.

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Klientská aplikace C, 64bitová, bez podprocesů

```
gcc -m64 -o amqsputc_64 amqsput0.c -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64 -lmqic
```

C client application, 64-bit, threaded

```
gcc -m64 -o amqsputc_64_r amqsput0.c -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64 -lmqic_r
-lpthread
```

Serverová aplikace C, 64bitová, bez podprocesů

```
gcc -m64 -o amqsput_64 amqsput0.c -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64 -lmqm
```

C server application, 64-bit, threaded

```
gcc -m64 -o amqsput_64_r amqsput0.c -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64 -lmqm_r
-lpthread
```

Klientská aplikace C + +, 64bitová, bez podprocesů

```
g++ -m64 -fsigned-char -o imqsputc_64 imqsput.cpp
-I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqc23gl -limqb23gl -lmqic
```

Klientská aplikace C + +, 64bitová verze

```
g++ -m64 -fsigned-char -o imqsputc_64_r imqsput.cpp
-I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqc23gl_r -limqb23gl_r -lmqic_r -lpthread
```

Serverová aplikace C + +, 64bitová, bez podprocesů

```
g++ -m64 -fsigned-char -o imqsput_64 imqsput.cpp
-I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -Wl,-rpath= MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=/usr/lib64 -limqs23gl -limqb23gl -lmqm
```

Serverová aplikace C + +, 64bitová verze

```
g++ -m64 -fsigned-char -o imqsput_64_r imqsput.cpp
-I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -Wl,-rpath= MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=/usr/lib64 -limqs23gl_r -limqb23gl_r -lmqm_r -lpthread
```

C exit exit, 64-bit, bez podprocesů

```
gcc -m64 -shared -fPIC -o /var/mqm/exits64/cliexit_64 cliexit.c
-I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=/usr/lib64 -lmqic
```

C exit exit, 64-bit, threaded

```
gcc -m64 -shared -fPIC -o /var/mqm/exits64/cliexit_64_r cliexit.c
-I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=/usr/lib64 -lmqic_r -lpthread
```

C server exit, 64-bit, bez podprocesů

```
gcc -m64 -shared -fPIC -o /var/mqm/exits64/srvexit_64 srvexit.c
-I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=/usr/lib64 -lmqm
```

C server exit, 64-bit, threaded

```
gcc -m64 -shared -fPIC -o /var/mqm/exits64/srvexit_64_r srvexit.c
```

```
-I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=/usr/lib64 -lmqm_r -lpthread
```

Linux *Příprava programů COBOL v produktu Linux*

Informace o přípravě programů v jazyce COBOL v produktu Linux a přípravě programů v jazyce COBOL pomocí programu IBM COBOL for Linux na platformě x86 a Micro Focus COBOL.

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

1. 32bitové příručky pro kopírování v COBOLu jsou instalovány v následujícím adresáři:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy32
   ```

   a symbolické odkazy jsou vytvořeny v:

   ```
   MQ_INSTALLATION_PATH/inc
   ```

2. Na 64bitových platformách jsou 64bitové knihy pro kopírování COBOL instalovány v následujícím adresáři:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy64
   ```

3. V následujících příkladech nastavte COBCPY na:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy32
   ```

   pro 32bitové aplikace a:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy64
   ```

   pro 64bitové aplikace.

Musíte propojit svůj program s jednou z následujících možností:

Soubor knihovny	Typ programu/ukončení
libmqmcb.so	Server pro COBOL
libmqicb.so	Klient pro COBOL
libmqmcb_r.so	Server pro COBOL (podprocesová aplikace)
libmqicb_r.so	Klient pro COBOL (podprocesová aplikace)

Příprava programů v jazyce COBOL pomocí programu IBM COBOL for Linux na platformě x86

Ukázkové programy COBOL jsou dodávány s produktem IBM MQ. Chcete-li zkompilovat takový program, zadejte příslušný příkaz z následujícího seznamu:

32bitová serverová aplikace bez podprocesů

```
$ cob2 -o amq0put0 amq0put0.cbl -q"BINARY(BE)" -q"FLOAT(BE)" -q"UTF16(BE)"
-L MQ_INSTALLATION_PATH/lib -lmqmcb -ICOBCPY_VALUE
```

32bitovou aplikaci klienta bez podprocesů

```
$ cob2 -o amq0put0 amq0put0.cbl -q"BINARY(BE)" -q"FLOAT(BE)" -q"UTF16(BE)"
-L MQ_INSTALLATION_PATH/lib -lmqicb -ICOBCPY_VALUE
```

32bitová serverová aplikace s podporou podprocesů

```
$ cob2_r -o amq0put0 amq0put0.cbl -q"BINARY(BE)" -q"FLOAT(BE)" -q"UTF16(BE)"
-qTHREAD -L MQ_INSTALLATION_PATH/lib -lmqmcb_r -ICOBCPY_VALUE
```

32bitovou aplikaci klienta s podprocesy

```
$ cob2_r -o amq0put0 amq0put0.cbl -q"BINARY(BE)" -q"FLOAT(BE)" -q"UTF16(BE)"
-qTHREAD -L MQ_INSTALLATION_PATH/lib -lmqicb_r -ICOBCPY_VALUE
```

Příprava programů v jazyce COBOL pomocí Micro Focus COBOL

Před kompilací programu nastavte proměnné prostředí následujícím způsobem:

```
export COBCPY=COBCPY_VALUE
export LIB= MQ_INSTALLATION_PATH lib:$LIB
```

Chcete-li zkompilovat 32bitový program COBOL, je-li podporován, použijte Micro Focus COBOL, zadejte:

```
$ cob32 -xvP amqsput.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb Server for COBOL
$ cob32 -xvP amqsput.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb Client for COBOL
$ cob32 -xtvP amqsput.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb_r Threaded Server for COBOL
$ cob32 -xtvP amqsput.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb_r Threaded Client for COBOL
```

Chcete-li zkompilovat 64bitový program COBOL pomocí Micro Focus COBOL, zadejte:

```
$ cob64 -xvP amqsput.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqmcb Server for COBOL
$ cob64 -xvP amqsput.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqicb Client for COBOL
$ cob64 -xtvP amqsput.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqmcb_r Threaded Server for COBOL
$ cob64 -xtvP amqsput.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqicb_r Threaded Client for COBOL
```

kde `amqsput` je ukázkový program.

Popis proměnných prostředí, které potřebujete, najdete v dokumentaci Micro Focus COBOL.

IBM i Sestavuje se vaše procedurální aplikace na IBM i

Publikace IBM i popisují, jak sestavit spustitelné aplikace z programů, které zapíšete, a které jsou spouštěny s produktem IBM i v systémech iSeries nebo System i .

Toto téma popisuje další úlohy a změny standardních úloh, které je třeba provést při sestavování procedurálních aplikací produktu IBM MQ for IBM i pro spouštění v systémech IBM i . Jsou podporovány programovací jazyky COBOL, C, C + +, Java a RPG. Informace o přípravě programů C + +, viz Použití C++. Informace o přípravě vašich programů Java najdete v tématu Použití modulu IBM MQ classes for Java.

Úlohy, které je třeba provést při vytváření spustitelné aplikace IBM MQ for IBM i , závisí na jazyku programování, ve kterém je zdrojový kód zapsán. Kromě kódování volání MQI ve vašem zdrojovém kódu musíte přidat příslušné jazykové příkazy, aby byly zahrnuty soubory definic dat produktu IBM MQ for IBM i pro jazyk, který používáte. Seznamte se s obsahem těchto souborů. Úplný popis najdete v tématu “Soubory definic dat produktu IBM MQ” na stránce 686 .

Příprava programů jazyka C v produktu IBM i

Produkt IBM MQ for IBM i podporuje zprávy o velikosti až 100 MB. Aplikační programy napsané v ILE C, které podporují IBM MQ zprávy větší než 16 MB, potřebují použít volbu kompilátoru *Teraprostor* k alokaci dostatečné paměti pro tyto zprávy.

Další informace o volbách kompilátoru jazyka C naleznete v příručce *WebSphere Development Studio ILE C/C++ Programmer's Guide*.

Chcete-li kompilovat modul C, můžete použít příkaz IBM i ,CRTCOD. Ujistěte se, že knihovna obsahující soubory začlenění (QMQM) je v seznamu knihoven, když kompilujete.

Potom musíte vytvořit vazbu výstupu kompilátoru s servisním programem pomocí příkazu CRTPGM.

Příklad příkazu pro prostředí bez podprocesů je:

Tabulka 132. Příklad CRTPGM v prostředí bez podprocesů	
Příkaz	**Typ programu/ukončení**
`CRTPGM PGM( pgmname ) MODULE( pgmname )` `BNDSRVPGM(QMQM/LIBMQM)`	Server nebo klient pro C

kde *pgmname* je jméno vašeho programu.

Příklad příkazu pro prostředí s vláknem je:

Tabulka 133. Příklad CRTPGM v prostředí vláken	
Příkaz	**Typ programu/ukončení**
`CRTPGM PGM( pgmname ) MODULE( pgmname )` `BNDSRVPGM(QMQM/LIBMQM_R)`	Server nebo klient pro C

kde *pgmname* je jméno vašeho programu.

V následujících tabulkách jsou uvedeny knihovny, které jsou potřebné při přípravě programů C na systému IBM i v prostředí bez podprocesů a v prostředí podprocesů.

Tabulka 134. Prostředí bez podprocesů	
Soubor knihovny	**Typ programu/ukončení**
LIBMQM	Server pro C
LIBMQIC & LIBMQM	Klient pro C

Tabulka 135. Prostředí s podprocesy	
Soubor knihovny	**Typ programu/ukončení**
LIBMQM_R	Server pro C
LIBMQIC_R & LIBMQM_R	Klient pro C

IBM i *Příprava programů COBOL v produktu IBM i*

Informace o přípravě programů v jazyce COBOL v produktu IBM i a o metodě přístupu k rozhraní MQI v rámci programu v jazyce COBOL.

Informace o této úloze

Pro přístup k rozhraní MQI z programů v jazyce COBOL poskytuje produkt IBM MQ for IBM i vázané procedurálního rozhraní volání poskytovaných servisními programy. Poskytuje přístup ke všem funkcím MQI v produktu IBM MQ for IBM ia podpora pro aplikace s podporou podprocesů. Toto rozhraní může být použito pouze s kompilátorem ILE COBOL.

Pro přístup k funkcím MQI se používá standardní syntaxe příkazu COBOL CALL.

Soubory kopií jazyka COBOL obsahující pojmenované konstanty a definice struktury pro použití s produktem MQI jsou obsaženy ve zdrojovém fyzickém souboru QMQM/QCBLLESRC.

Soubory kopie v jazyce COBOL používají jako oddělovač řetězců znak apostrofu ('). Kompilátory jazyka COBOL produktu IBM i předpokládají, že oddělovačem je uvozovka ("). Chcete-li zabránit kompilátorům generování varovných zpráv, zadejte `OPTION (*APOST)` na příkazy **CRTCBLPGM**, **CRTBNDCBL**nebo **CRTCBLMOD**.

Chcete-li, aby kompilátor akceptoval znak jednoduché uvozovky (') jako oddělovač řetězce v souborech kopií v jazyce COBOL, použijte volbu kompilátoru \APOST.

Poznámka: Rozhraní dynamického volání není v produktu IBM MQ 9.0k dispozici.

Chcete-li použít rozhraní volání procedury vazby, proveďte následující kroky.

Postup

1. Vytvořte modul pomocí kompilátoru **CRTCBLMOD** specifikující tento parametr:

```
LINKLIT(*PRC)
```

2. Použijte příkaz **CRTPGM** k vytvoření objektu programu a uveďte odpovídající parametr:

   Pro aplikace bez podprocesů:

```
BNDSRVPGM(QMQM/AMQ0STUB)        Server for COBOL for non-threaded applications
BNDSRVPGM(QMQM/AMQCSTUB)        Client for COBOL for non-threaded applications
```

   Pro aplikace se závitem:

```
BNDSRVPGM(QMQM/AMQ0STUB_R)        Server for COBOL for threaded applications
BNDSRVPGM(QMQM/AMQCSTUB_R)        Client for COBOL for threaded applications
```

   Poznámka: S výjimkou programů vytvořených pomocí kompilátoru V4R4 ILE COBOL a obsahujících volbu THREAD (SERIALIZE) v příkazu PROCESS nesmí programy v jazyce COBOL používat knihovny IBM MQ s podporou podprocesů. Dokonce i v případě, že byl program v jazyce COBOL bezpečným způsobem zabezpečeným způsobem, buďte opatrní při návrhu aplikace, protože THREAD (SERIALIZE) vynutí serializaci procedur jazyka COBOL na úrovni modulu a může ovlivnit celkový výkon.

   Viz příručka *WebSphere Development Studio: ILE COBOL Programmer's Guide* a *WebSphere Development Studio: ILE COBOL Reference* , kde získáte další informace.

   Další informace o kompilaci aplikace CICS naleznete v příručce *CICS for IBM i Application Programming Guide*, SC41-5454.

Příprava programů produktu CICS v produktu IBM i

Zde se dozvíte o krocích nezbytných při přípravě programů CICS v produktu IBM i.

Chcete-li vytvořit program, který obsahuje příkazy EXEC CICS a volání MQI, proveďte následující kroky:

1. Je-li to nutné, připravte mapy pomocí příkazu CRTCICSMAP.
2. Přeložte příkazy EXEC CICS do příkazů nativního jazyka. Použijte příkaz CRTCICSC pro program C. Použijte příkaz CRTCICSCBL pro program COBOL.

   Začlenit `CICSOPT(*NOGEN)` do příkazu CRTCICSC nebo CRTCICSCBL. Tím se zastaví zpracování, aby bylo možné zahrnout příslušné servisní programy CICS a IBM MQ . Tento příkaz umístí kód do QTEMP/ QACYCICSjako výchozí.
3. Zkompilujte zdrojový kód pomocí příkazu CRTCMOD (pro program v jazyce C) nebo příkaz CRTCBLMOD (pro program v jazyce COBOL).
4. Použijte CRTPGM k propojení kompilovaného kódu s odpovídajícími servisními programy CICS a IBM MQ . Tím se vytvoří spustitelný program.

Zde je uveden příklad takového kódu (zkompiluje dodaný ukázkový program CICS):

```
CRTCICSC  OBJ(QTEMP/AMQSCIC0) SRCFILE(/MQSAMP/QCSRC) +
     SRCMBR(AMQSCIC0) OUTPUT(*PRINT) +
     CICSOPT(*SOURCE *NOGEN)
CRTCMOD   MODULE(MQTEST/AMQSCIC0) +
     SRCFILE(QTEMP/QACYCICS) OUTPUT(*PRINT)
```

```
CRTPGM   PGM(MQTEST/AMQSCIC0) MODULE(MQTEST/AMQSCIC0) +
     BNDSRVPGM(QMQM/LIBMQIC QCICS/AEGEIPGM)
```

IBM i *Příprava programů RPG v produktu IBM i*

Používáte-li produkt IBM MQ for IBM i, můžete psát své aplikace v jazyce RPG.

Další informace viz “Kódování programů IBM MQ v jazyce RPG (pouzeIBM i)” na stránce 1033a prostudujte si publikaci IBM i Application Programming Reference (ILE/RPG).

Pokyny k programování SQL

Zde se dozvíte o krocích nutných při sestavování aplikace v systému IBM i pomocí jazyka SQL.

Pokud váš program obsahuje příkazy EXEC SQL a volání MQI, proveďte tyto kroky:

1. Přeložte příkazy EXEC SQL do příkazů nativního jazyka. Pro program v jazyce C použijte příkaz CRTSQLCI. Použijte příkaz CRTSQLCBLI pro program v jazyce COBOL.

   Zahrnout `OPTION(*NOGEN)` do příkazu CRTSQLCI nebo CRTSQLCBLI. Tím se zastaví zpracování, aby bylo možné zahrnout příslušné servisní programy produktu IBM MQ . Tento příkaz umístí kód do QTEMP/QSQLTEMP do výchozího stavu.
2. Zkompilujte zdrojový kód pomocí příkazu CRTCMOD (pro program v jazyce C) nebo příkaz CRTCBLMOD (pro program v jazyce COBOL).
3. Použijte CRTPGM k propojení kompilovaného kódu s odpovídajícími servisními programy IBM MQ . Tím se vytvoří spustitelný program.

Příklad takového kódu následuje (zkompiluje program, SQLTEST, v knihovně, SQLUSER):

```
CRTSQLCI  OBJ(MQTEST/SQLTEST) SRCFILE(SQLUSER/QCSRC) +
     SRCMBR(SQLTEST) OUTPUT(*PRINT) OPTION(*NOGEN)
CRTCMOD   MODULE(MQTEST/SQLTEST) +
     SRCFILE(QTEMP/QSQLTEMP) OUTPUT(*PRINT)
CRTPGM   PGM(MQTEST/SQLTEST) +
     BNDSRVPGM(QMQM/LIBMQIC)
```

Solaris Sestavuje se vaše procedurální aplikace na Solaris

Tyto informace popisují další úlohy a změny standardních úloh, které musíte provést při sestavování aplikací produktu IBM MQ for Solaris pro spouštění v produktu Solaris.

Jsou podporovány programovací jazyky COBOL, C a C++. Informace o přípravě programů C + +, viz Použití C++.

Kromě kódování volání MQI ve vašem zdrojovém kódu je třeba přidat příslušné soubory začlenění. Seznamte se s obsahem těchto souborů. Úplný popis najdete v tématu “Soubory definic dat produktu IBM MQ” na stránce 686 .

V celém tomto tématu se znak zpětného lomítka (\) používá k rozdělení dlouhých příkazů na více než jeden řádek. Nezadávejte tento znak, zadejte každý příkaz jako jednu řádku.

Příprava programů jazyka C v produktu Solaris

Předkompilované C programy jsou dodávány v adresáři *MQ_INSTALLATION_PATH*`/samp/bin` .

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Další informace o programování 64bitových aplikací najdete v tématu Koding standardů na 64bitových platformách.

Chcete-li používat programy na počítači, který má nainstalované pouze IBM MQ MQI client for Solaris , zkompilujte programy a propojte je s knihovnou klienta (`-lmqic`).

Pokud použijete nepodporovaný kompilátor `/usr/ucb/cc`, může se vaše aplikace zkompilovat a úspěšně propojit. Když však spustíte aplikaci, nezdaří se, když se pokusí připojit ke správci front.

Poznámka: 32bitoví klienti Solaris x86 s protokolem SSL a TLS, nakonfigurovaní pro operace dle standardu FIPS 140-2, selžou, budete-li je provozovat na systémech Intel. K tomuto selhání dochází, protože soubor 32bitové knihovny GSKit-Crypto Solaris x86 vyhovující specifikaci FIPS 140-2 se do čipové sady Intel nenačte. V zasažených systémech se do protokolu chyb klienta nahlásí chyba AMQ9655. Tento problém vyřešíte tak, že vypnete kompatibilitu se specifikací FIPS 140-2, nebo znovu zkompilujete aplikaci klienta pro 64-bit bitů, protože 64bitový kód není dotčen.

Propojování knihoven

Musíte propojit s knihovnami produktu IBM MQ , které jsou vhodné pro daný typ aplikace:

Soubory knihovny	Typ programu/ukončení
libmqm.so	Server pro C
libmqic.so & libmqm.so	Klient pro C

Poznámka:

1. Pokud píšete instalovatelnou službu (další informace viz Správa), propojte se s knihovnou `libmqmzf.so` .
2. Pokud vytváříte aplikaci pro externí koordinaci správce transakcí kompatibilní se standardem XA, jako např. IBM TXSeries Encina nebo BEA Tuxedo, musíte se připojit k serveru `libmqmxa.so` (nebo `libmqmxa64.so` , pokud váš správce transakcí zachází s typem " long ' typu 64 bit) a s knihovnami `libmqz.so` .
3. Musíte propojit knihovny produktu IBM MQ před všemi ostatními knihovnami produktu.

Sestavování aplikací v systému x86-64
Toto téma obsahuje příklady příkazů používaných k sestavování programů v různých prostředích na platformě x86-64 .

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Klientská aplikace C, 32bitová

```
cc -xarch=386 -mt -o amqsputc_32 amqsput0.c -I MQ_INSTALLATION_PATH/inc -L
MQ_INSTALLATION_PATH/lib
-R MQ_INSTALLATION_PATH/lib -R/usr/lib/32 -lmqic -lsocket -lnsl -ldl
```

Klientská aplikace C, 64bitová

```
cc -xarch=amd64 -mt -o amqsputc_64 amqsput0.c -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -R MQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -lmqic -lsocket
-lnsl -ldl
```

Serverová aplikace C, 32 bitů

```
cc -xarch=386 -mt -o amqsput_32 amqsput0.c -I MQ_INSTALLATION_PATH/inc -L
MQ_INSTALLATION_PATH/lib
-R MQ_INSTALLATION_PATH/lib -R/usr/lib/32 -lmqm -lsocket -lnsl -ldl
```

Serverová aplikace C, 64bitová

```
cc -xarch=amd64 -mt -o amqsput_64 amqsput0.c -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -R MQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -lmqm -lsocket
-lnsl -ldl
```

C++ client application, 32-bit

```
CC -xarch=386 -mt -o imqsputc_32 imqsput.cpp -I MQ_INSTALLATION_PATH/inc -L
MQ_INSTALLATION_PATH/lib
-R MQ_INSTALLATION_PATH/lib -R/usr/lib/32 -limqc23as -limqb23as -lmqic -lsocket -lnsl -ldl
```

Klientská aplikace C + +, 64bitová

```
CC -xarch=amd64 -mt -o imqsputc_64 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -R MQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -limqc23as
-limqb23as
-lmqic -lsocket -lnsl -ldl
```

Serverová aplikace C + +, 32bitová

```
CC -xarch=386 -mt -o imqsput_32 imqsput.cpp -I MQ_INSTALLATION_PATH/inc -L
MQ_INSTALLATION_PATH/lib
-R MQ_INSTALLATION_PATH/lib -R/usr/lib/32 -limqs23as -limqb23as -lmqm
-lsocket -lnsl -ldl
```

Serverová aplikace C + +, 64bitový

```
CC -xarch=amd64 -mt -o imqsput_64 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -R MQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -limqs23as
-limqb23as -lmqm
-lsocket -lnsl -ldl
```

C client exit, 32-bit

```
cc -xarch=386 -mt -G -KPIC -o /var/mqm/exits/cliexit_32 cliexit.c
-I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib -R MQ_INSTALLATION_PATH/lib
-R/usr/lib/32
-lmqic -lsocket -lnsl -ldl
```

C client exit, 64-bit

```
cc -xarch=amd64 -mt -G -KPIC -o /var/mqm/exits64/cliexit_64 cliexit.c
-I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib64 -R MQ_INSTALLATION_PATH/lib64
-R/usr/lib/64
-lmqic -lsocket -lnsl -ldl
```

Ukončení serveru C, 32 bitů

```
cc -xarch=386 -mt -G -KPIC -o /var/mqm/exits/srvexit_32 srvexit.c
-I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib -R MQ_INSTALLATION_PATH/lib
-R/usr/lib/32
-lmqm -lsocket -lnsl -ldl
```

Ukončení serveru C, 64bitový

```
cc -xarch=amd64 -mt -G -KPIC -o /var/mqm/exits64/srvexit_64 srvexit.c
-I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib64 -R MQ_INSTALLATION_PATH/lib64
-R/usr/lib/64
-lmqm -lsocket -lnsl -ldl
```

Sestavování aplikací na platformě SPARC
Toto téma obsahuje příklady příkazů používaných k sestavování programů v různých prostředích na platformě SPARC.

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Klientská aplikace C, 32bitová

```
cc -xarch=v8plus -mt -o amqsputc_32 amqsput0.c -I MQ_INSTALLATION_PATH/inc -L
MQ_INSTALLATION_PATH/lib
-R MQ_INSTALLATION_PATH/lib -R/usr/lib/32 -lmqic -lsocket -lnsl -ldl
```

Klientská aplikace C, 64bitová

```
cc -xarch=v9 -mt -o amqsputc_64 amqsput0.c -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -R MQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -lmqic
-lsocket -lnsl -ldl
```

Serverová aplikace C, 32 bitů

```
cc -xarch=v8plus -mt -o amqsput_32 amqsput0.c -I MQ_INSTALLATION_PATH/inc -L
MQ_INSTALLATION_PATH/lib
-R MQ_INSTALLATION_PATH/lib -R/usr/lib/32 -lmqm -lsocket -lnsl -ldl
```

Serverová aplikace C, 64bitová

```
cc -xarch=v9 -mt -o amqsput_64 amqsput0.c -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -R MQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -lmqm
-lsocket -lnsl -ldl
```

C++ client application, 32-bit

```
CC -xarch=v8plus -mt -o imqsputc_32 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -R MQ_INSTALLATION_PATH/lib -R/usr/lib/32 -limqc23as -limqb23as
-lmqic
-lsocket -lnsl -ldl
```

Klientská aplikace C + +, 64bitová

```
CC -xarch=v9 -mt -o imqsputc_64 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -R MQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -limqc23as
-limqb23as
-lmqic -lsocket -lnsl -ldl
```

Serverová aplikace C + +, 32bitová

```
CC -xarch=v8plus -mt -o imqsput_32 imqsput.cpp -I MQ_INSTALLATION_PATH/inc -L
MQ_INSTALLATION_PATH/lib
-R MQ_INSTALLATION_PATH/lib -R/usr/lib/32 -limqs23as -limqb23as -lmqm
-lsocket -lnsl -ldl
```

Serverová aplikace C + +, 64bitový

```
CC -xarch=v9 -mt -o imqsput_64 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -R MQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -limqs23as
-limqb23as -lmqm
-lsocket -lnsl -ldl
```

C client exit, 32-bit

```
cc -xarch=v8plus -mt -G -KPIC -o /var/mqm/exits/cliexit_32 cliexit.c
-I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib -R MQ_INSTALLATION_PATH/lib
-R/usr/lib/32
-lmqic -lsocket -lnsl -ldl
```

C client exit, 64-bit

```
cc -xarch=v9 -mt -G -KPIC -o /var/mqm/exits64/cliexit_64 cliexit.c
```

```
-I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib64 -R MQ_INSTALLATION_PATH/lib64
-R/usr/lib/64
-lmqic -lsocket -lnsl -ldl
```

Ukončení serveru C, 32 bitů

```
cc -xarch=v8plus -mt -G -KPIC -o /var/mqm/exits/srvexit_32 srvexit.c
-I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib -R MQ_INSTALLATION_PATH/lib
-R/usr/lib/32
-lmqm -lsocket -lnsl -ldl
```

Ukončení serveru C, 64bitový

```
cc -xarch=v9 -mt -G -KPIC -o /var/mqm/exits64/srvexit_64 srvexit.c
-I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib64 -R MQ_INSTALLATION_PATH/lib64
-R/usr/lib/64
-lmqm -lsocket -lnsl -ldl
```

Příprava programů COBOL v produktu Solaris

Informace o přípravě programů v jazyce COBOL v produktu Solaris.

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

1. 32bitové příručky pro kopírování v COBOLu jsou instalovány v následujícím adresáři:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy32
   ```

   a symbolické odkazy jsou vytvořeny v:

   ```
   MQ_INSTALLATION_PATH/inc
   ```

2. 64bitové příručky pro kopírování v jazyce COBOL jsou instalovány v následujícím adresáři:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy64
   ```

3. V následujících příkladech nastavte COBCPY na:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy32
   ```

   pro 32bitové aplikace a:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy64
   ```

   pro 64bitové aplikace.

Proveďte kompilaci programů pomocí kompilátoru Micro Focus. Kopírovaté soubory, které deklarují struktury, jsou v *MQ_INSTALLATION_PATH*/inc:

```
$ export LIB= MQ_INSTALLATION_PATH/lib:$LIB
$ export COBCPY="COBCPY_VALUE"
```

Kompilace 32bitových programů:

- `$ cob32 -xv amqs0put0.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb`

  Server pro COBOL
- `$ cob32 -xv amqs0put0.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb`

  Klient pro COBOL
- `$ cob32 -xtv amqs0put0.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb_r`

Threadovaný server pro COBOL

- `$ cob32 -xtv amqs0put0.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb_r`

  Klient s podporou podprocesů pro COBOL

Kompilace 64-bitových programů:

- `$ cob64 -xv amqs0put0.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqmcb`

  Server pro COBOL
- `$ cob64 -xv amqs0put0.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqicb`

  Klient pro COBOL
- `$ cob64 -xtv amqs0put0.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqmcb_r`

  Threadovaný server pro COBOL
- `$ cob64 -xtv amqs0put0.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqicb_r`

  Klient s podporou podprocesů pro COBOL

kde *amqs0put0.cbl* je ukázkový program.

Musíte propojit svůj program s jednou z následujících možností:

- libmqmcb.so

  Server pro COBOL
- libmqicb.so

  Klient pro COBOL

Příprava programů produktu CICS v produktu Solaris

Informace o přípravě programů produktu CICS v produktu Solaris.

Je k dispozici modul přepínače XA, který vám umožňuje propojení CICS s IBM MQ:

Tabulka 136. Základní kód pro aplikace produktu CICS (Solaris)

Popis	**C (zdroj)**	**C (exec)**
inicializační rutina XA	amqzscix.c	amqzsc- položky TXSeries pro Solaris

Vždy propojte své transakce s bezpečnou knihovnou IBM MQ libmqm.so.

Další informace o podpoře CICS transakcí v příručce Správanajdete.

Podpora produktu TXSeriesCICS

Produkt IBM MQ for Solaris podporuje prostředí TXSeries CICS pomocí rozhraní XA.

Zapište IBM MQ programy, které jsou zavedeny do stejné oblasti CICS buď v jazyce C, nebo v COBOLu. Nemůžete vytvořit kombinaci volání C a COBOL MQI do stejné oblasti CICS . Většina volání MQI ve druhém použitém jazyce se nezdařila s kódem příčiny `MQRC_HOBJ_ERROR`.

Příprava programů jazyka COBOL pro produkt CICS pomocí Micro Focus COBOL

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Chcete-li použít funkci Micro Focus COBOL, postupujte takto:

1. Přidejte modul knihovny běhové komponenty produktu IBM MQ COBOL do knihovny běhového prostředí pomocí následujícího příkazu:

```
cicsmkcobol -L/usr/lib/dce -L MQ_INSTALLATION_PATH/lib \
            MQ_INSTALLATION_PATH/lib/libmqmcbrt.o -lmqe
```

   Poznámka: With `cicsmkcobol`, IBM MQ does not allow you to make MQI calls in the C programming language from your COBOL application.

   Mají-li existující aplikace taková volání, přesuňte tyto funkce z aplikací v jazyce COBOL do vaší vlastní knihovny, například `myMQ.so`. Po přesunu těchto funkcí nezahrnujte knihovnu IBM MQ `libmqmcbrt.o` při sestavování aplikace v jazyce COBOL pro produkt CICS.

   Navíc, pokud vaše aplikace v jazyku COBOL nevolá žádné volání COBOL MQI, nepropojujte `libmqmz_r` s `cicsmkcobol`.

   Tím se vytvoří soubor metod jazyka COBOL Micro Focus a povolí knihovně jazyka COBOL pro běhové prostředí produktu CICS volání IBM MQ v systémech UNIX and Linux .

   Poznámka: Spusťte produkt `cicsmkcobol` pouze při instalaci jednoho z následujících produktů:

   - Nová verze nebo vydání Micro Focus COBOL
   - Nová verze nebo vydání TXSeries pro Solaris
   - Nová verze nebo vydání všech podporovaných databázových produktů (pouze pro transakce v jazyce COBOL)
   - Nová verze nebo vydání IBM MQ

2. Exportovat následující proměnnou prostředí:

```
COBCPY= MQ_INSTALLATION_PATH/inc export COBCPY
```

3. Přeložte, zkompilujte a propojte program zadáním příkazu:

```
cicstcl -l COBOL -e yourprog.ccp
```

Příprava programů v programu CICS C

Sestavte programy CICS C pomocí standardních zařízení produktu CICS :

1. Exportujte ***jeden*** z následujících proměnných prostředí:
   - LDFLAGS = "-L *MQ_INSTALLATION_PATH* l lib -lmqm_r" export LDFLAGS
   - USERLIB = "-L *MQ_INSTALLATION_PATH* l lib -lmqm_r" export USERLIB
2. Přeložte, zkompilujte a propojte program zadáním příkazu:

```
cicstcl -l C amqscic0.ccs
```

CICS vzorová transakce C

Ukázka zdroje C pro transakci produktu CICS IBM MQ je k dispozici v rámci AMQSCIC0.CCS. Transakce čte zprávy z přenosové fronty SYSTEM.SAMPLE.CICS.WORKQUEUE na výchozím správci front a umisťuje je do lokální fronty s názvem fronty, který je obsažen v záhlaví přenosu zprávy. Veškerá selhání se odesílají do fronty SYSTEM.SAMPLE.CICS.DLQ. Použijte ukázkový skript MQSC AMQSCIC0.TST pro vytvoření těchto front a ukázkových vstupních front.

Windows Sestavuje se vaše procedurální aplikace na Windows

Publikace Windows Systems Publications popisují, jak vytvářet spustitelné aplikace z programů, které napíšete.

Toto téma popisuje další úlohy a změny standardních úloh, které musíte provést při sestavování aplikací IBM MQ for Windows pro spouštění v systému Windows . Jsou podporovány programovací jazyky ActiveX, C, C + +, COBOL a Visual Basic. Informace o přípravě programů ActiveX viz Použití rozhraní modelu objektu komponenty (WebSphere MQ Automation Classes for ActiveX). Informace o přípravě programů C + +, viz Použití C++.

Úlohy, které musíte provést, abyste vytvořili spustitelnou aplikaci pomocí produktu IBM MQ for Windows , se liší pomocí programovacího jazyka, ve kterém je váš zdrojový kód zapsán. Kromě kódování volání MQI ve vašem zdrojovém kódu musíte přidat příslušné jazykové příkazy, aby bylo možné zahrnout soubory začlenění produktu IBM MQ for Windows do jazyka, který používáte. Seznamte se s obsahem těchto souborů. Úplný popis najdete v tématu “Soubory definic dat produktu IBM MQ” na stránce 686 .

Sestavování 64bitových aplikací v systému Windows

V produktu IBM MQ for Windowsjsou podporovány 32bitové i 64bitové aplikace. Spustitelné soubory a soubory knihovny produktu IBM MQ jsou dodávány v 32bitovém i 64bitovém formuláři, používejte příslušnou verzi v závislosti na aplikaci, se kterou pracujete.

Spustitelné soubory a knihovny

32bitové i 64bitové verze knihoven produktu IBM MQ jsou dodávány v následujících umístěních:

Tabulka 137. Umístění knihoven IBM MQ

Verze knihovny	**Adresář obsahující soubory knihovny**
32bitové	*MQ_INSTALLATION_PATH* \Tools\Lib
64bitová	*MQ_INSTALLATION_PATH* \Tools\Lib64

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

32bitové aplikace pokračují v práci normálně po migraci. 32 bitové soubory existují ve stejném adresáři jako v předchozích verzích produktu.

Chcete-li vytvořit 64bitovou verzi, musíte se ujistit, že je vaše prostředí nakonfigurováno pro použití souborů knihovny v produktu *MQ_INSTALLATION_PATH* \Tools\Lib64. Ujistěte se, že proměnná prostředí LIB není nastavena, aby se podíval do složky obsahující 32bitové knihovny.

Příprava programů jazyka C v produktu Windows

Pracujte v typickém prostředí produktu Windows ; IBM MQ for Windows nevyžaduje nic speciálního.

Další informace o programování 64bitových aplikací najdete v tématu Kodové standardy na 64bitových platformách.

- Propojte své programy s odpovídajícími knihovnami poskytnutými serverem IBM MQ:

Soubor knihovny	**Typ programu/ukončení**
MQ_INSTALLATION_PATH \Tools\Lib\mqm.lib	server pro 32bitovou verzi C
MQ_INSTALLATION_PATH \Tools\Lib\mqic.lib	klient pro 32bitovou verzi C
MQ_INSTALLATION_PATH \Tools\Lib\mqicxa.lib	klient pro 32bitovou transakci s koordinačním transakcí

Soubor knihovny	Typ programu/ukončení
MQ_INSTALLATION_PATH\Tools\Lib64\mqm.lib	server pro 64 bitů C
MQ_INSTALLATION_PATH\Tools\Lib64\mqic.lib	klient pro 64 bitů C
MQ_INSTALLATION_PATH\Tools\Lib64\mqicxa.lib	klient pro 64bitový C s koordinací transakcí

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Následující příkaz poskytuje příklad kompilace ukázkového programu amqsget0 (s použitím kompilátoru Microsoft Visual C + +).

Pro 32 bitové aplikace:

```
cl -MD amqsget0.c -Feamqsget.exe MQ_INSTALLATION_PATH\Tools\Lib\mqm.lib
```

Pro 64bitové aplikace:

```
cl -MD amqsget0.c -Feamqsget.exe MQ_INSTALLATION_PATH\Tools\Lib64\mqm.lib
```

Poznámka:

- Pokud zapisujete instalovatelnou službu (další informace naleznete v příručce Správa), je třeba vytvořit odkaz na knihovnu mqmzf.lib .
- Pokud vytváříte aplikaci pro externí koordinaci správce transakcí kompatibilní se standardem XA, jako např. IBM TXSeries Encina nebo BEA Tuxedo, musíte vytvořit odkaz na knihovnu mqmxa.lib nebo mqmxa.lib .
- Pokud vytváříte uživatelskou proceduru produktu CICS , přejděte do knihovny mqmcics4.lib .
- Musíte propojit knihovny produktu IBM MQ před všemi ostatními knihovnami produktu.

- Knihovny DLL se musí nacházet v cestě (PATH), kterou jste zadali.
- Pokud použijete malá písmena vždy, když je to možné, můžete se přesunout z IBM MQ for Windows do IBM MQ na systémech UNIX and Linux , kde je použití malých písmen nutné.

Příprava programů serveru CICS a transakčních serverů

Ukázka zdroje C pro transakci produktu CICS IBM MQ je k dispozici v rámci AMQSCIC0.CCS. Sestavujete ji pomocí standardních mechanizmů produktu CICS . Například pro TXSeries for Windows 2000:

1. Nastavte proměnnou prostředí (zadejte na jednom řádku následující kód):

   ```
   set CICS_IBMC_FLAGS=-I MQ_INSTALLATION_PATH\Tools\C\Include;
   %CICS_IBMC_FLAGS%
   ```

2. Nastavte proměnnou prostředí USERLIB:

   ```
   set USERLIB=MQM.LIB;%USERLIB%
   ```

3. Translate, compile, and link the sample program:

```
cicstcl -l IBMC amqscic0.ccs
```

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

To je popsáno v příručce *Transaction Server for Windows NT Application Programming Guide (CICS) V4*.

Další informace o podpoře CICS transakcí v příručce Správanajdete.

Windows *Příprava programů COBOL v produktu Windows*

Tyto informace použijte k seznámení se s programy v jazyce COBOL v produktu Windowsa k přípravě programů CICS a transakčních programů.

1. 32bitové příručky pro kopírování v COBOLu jsou instalovány v následujícím adresáři: *MQ_INSTALLATION_PATH* \Tools\cobol\CopyBook.
2. 64bitové příručky pro COBOL jsou instalovány v následujícím adresáři: *MQ_INSTALLATION_PATH* \Tools\cobol\CopyBook64
3. V následujících příkladech nastavte CopyBook na:

   ```
   CopyBook
   ```

   pro 32bitové aplikace a:

   ```
   CopyBook64
   ```

   pro 64bitové aplikace.

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Chcete-li připravit programy v jazyce COBOL na systémech Windows , spojte svůj program s jednou z následujících knihoven poskytnutých produktem IBM MQ:

Soubor knihovny	Typ programu nebo ukončení
MQ_INSTALLATION_PATH \Tools\Lib\mqmcb	32bitový server pro Micro COBOL
MQ_INSTALLATION_PATH \Tools\Lib\mqiccb	32bitový klient pro Micro COBOL
MQ_INSTALLATION_PATH \Tools\Lib64\mqmcb	64bitový server pro Micro COBOL
MQ_INSTALLATION_PATH \Tools\Lib64\mqiccb	64bitový klient pro Micro COBOL

Pokud spouštíte program v prostředí klienta MQI, ujistěte se, že se knihovna DOSCALLS objeví před jakoukoli knihovnou COBOL nebo IBM MQ .

Příprava programů v jazyce COBOL pomocí Micro Focus COBOL

Znovu propojte všechny existující 32bitové programy IBM MQ Micro Focus, které používají buď mqmcb.lib nebo mqiccb.lib, a ne knihovny mqmcbb a mqiccbb .

Chcete-li kompilovat, například ukázkový program amq0put0, pomocí Micro Focus COBOL:

1. Nastavte proměnnou prostředí COBCPY tak, aby ukazovala na zakladače IBM MQ COBOL (zadejte na jednom řádku tento kód):

   ```
   set COBCPY= MQ_INSTALLATION_PATH\
   Tools\Cobol\Copybook
   ```

2. Kompilujte program, aby vám dal objektový soubor:

```
cobol amq0put0 LITLINK
```

3. Propojte soubor objektu se systémem běhového prostředí.
   - Nastavte proměnnou prostředí LIB tak, aby ukazovala na knihovny COBOL kompilátoru.
   - Propojte soubor objektu pro použití na serveru IBM MQ :

     ```
     cbllink amq0put0.obj mqmcb.lib
     ```

   - Nebo propojte soubor s objektem pro použití na klientovi IBM MQ :

     ```
     cbllink amq0put0.obj mqiccb.lib
     ```

Příprava programů serveru CICS a transakčních serverů

Chcete-li kompilovat a propojit TXSeries pro produkt Windows NT, program V5.1 pomocí IBM VisualAge COBOL:

1. Nastavte proměnnou prostředí (zadejte na jednom řádku následující kód):

   ```
   set CICS_IBMCOB_FLAGS= MQ_INSTALLATION_PATH\
   Cobol\Copybook\VAcobol;%CICS_IBMCOB_FLAGS%
   ```

2. Nastavte proměnnou prostředí USERLIB:

   ```
   set USERLIB=MQMCBB.LIB
   ```

3. Translate, compile, and link your program:

   ```
   cicstcl -l IBMCOB myprog.ccp
   ```

To je popsáno v příručce *Transaction Server for Windows NT, V4 Application Programming* Guide.

Chcete-li kompilovat a propojit CICS pro program Windows V5 pomocí Micro Focus COBOL, postupujte takto:

- Nastavte proměnnou INCLUDE:

  ```
  set
  INCLUDE=drive:\programname\ibm\websphere\tools\c\include;
          drive:\opt\cics\include;%INCLUDE%
  ```

- Nastavte proměnnou prostředí COBCPY:

  ```
  setCOBCPY=drive:\programname\ibm\websphere\tools\cobol\copybook;
            drive:\opt\cics\include
  ```

- Nastavte volby jazyka COBOL:
  - `set`
  - `COBOPTS=/LITLINK /NOTRUNC`

a spusťte následující kód:

```
cicstran cicsmq00.ccp
cobol cicsmq00.cbl /LITLINK /NOTRUNC
cbllink -D -Mcicsmq00 -Ocicsmq00.cbmfnt cicsmq00.obj
%CICSLIB%\cicsprCBMFNT.lib user32.lib msvcrt.lib kernel32.lib mqmcb.lib
```

Windows *Příprava programů produktu Visual Basic v produktu Windows*

Informace, které je třeba zvážit při používání programů produktu Microsoft Visual Basic v systému Windows.

V 9.0.0 V produktu IBM MQ 9.0je podpora produktu Microsoft Visual Basic 6.0 zamítnuta. Třídy IBM MQ pro .NET jsou doporučenými náhradními technologiemi. Další informace viz Vývoj aplikací .NET.

Poznámka: 64bitové verze souborů modulu Visual Basic nejsou dodány.

Příprava programu Visual Basic na systému Windows:

1. Vytvořit nový projekt
2. Přidejte dodaný soubor modulu, CMQB.BASdo projektu.
3. Pokud je potřebujete, přidejte další dodané soubory modulu:
   - CMQBB.BAS: podpora MQAI
   - CMQCFB.BAS: podpora PCF
   - CMQXB.BAS: Kanál ukončí podporu
   - CMQPSB.BAS: Publikování/odběr

Informace o použití volání MQCONNXAny z produktu Visual Basicnaleznete v příručce “Kódování v produktu Visual Basic” na stránce 1029 .

Před provedením jakýchkoli volání MQI v kódu projektu volejte proceduru MQ_SETDEFAULTS. Tato procedura nastaví výchozí struktury, které vyžaduje volání MQI.

Určete, zda vytváříte server nebo klienta produktu IBM MQ před kompilací nebo spuštěním projektu nastavením proměnné pro podmíněnou kompilaci *MqType*. Nastavte *MqType* v projektu Visual Basic na 1 pro server nebo 2 pro klienta následujícím způsobem:

1. Vyberte nabídku Projekt.
2. Vyberte položku *Name* Vlastnosti (kde *Name* je název aktuálního projektu).
3. Vyberte kartu Make v dialogovém okně.
4. V poli Argumenty podmíněné kompilace zadejte tento typ pro server:

   ```
   MqType=1
   ```

   nebo toto pro klienta:

   ```
   MqType=2
   ```

Související pojmy

“Kódování v produktu Visual Basic” na stránce 1029
Informace, které je třeba zvážit při kódování programů IBM MQ v produktu Microsoft Visual Basic. Produkt Visual Basic je podporován pouze v systému Windows.

Související odkazy

“Propojení aplikací produktu Visual Basic s kódem produktu IBM MQ MQI client” na stránce 882
You can link Microsoft Visual Basic applications with the IBM MQ MQI client code on Windows.

Uživatelská procedura zabezpečení SSPI

Produkt IBM MQ for Windows poskytuje uživatelskou proceduru zabezpečení pro server IBM MQ MQI client i pro server IBM MQ . Jedná se o program výstupního bodu kanálu, který poskytuje ověření pro kanály produktu IBM MQ pomocí rozhraní SSPI (Security Services Programming Interface). SSPI poskytuje integrované bezpečnostní mechanizmy systémů Windows .

Balíky zabezpečení jsou načteny z adresáře security.dll nebo secur32.dll. Tyto knihovny DLL se dodávají spolu s operačním systémem.

Jednosměrné ověření se poskytuje pomocí ověřovacích služeb NTLM. Dvojcestné ověření je poskytnuto pomocí ověřovacích služeb Kerberos .

Uživatelský program zabezpečení je dodáván ve zdrojovém formátu a ve formátu objektu. Kód objektu můžete použít tak, jak je, nebo můžete použít zdrojový kód jako výchozí bod k vytvoření svých vlastních uživatelských programů.

Další informace najdete v tématu "Použití uživatelské procedury zabezpečení SSPI v systému Windows" na stránce 1112.

Úvod do uživatelských procedur pro zabezpečení

Procedura zabezpečení vytvoří zabezpečené připojení mezi dvěma programy procedury zabezpečení. Jeden z těchto programů odpovídá odesílajícímu agentu MCA (message channel agent) a druhý přijímajícímu agentu MCA.

Program, který iniciuje zabezpečené připojení, tj. první program, který získá řízení po zřízení relace MCA, je znám jako *kontextový iniciátor*. Partnerský program je znám jako *služba kontextu kontextu*.

V následující tabulce jsou uvedeny některé typy kanálů, které jsou inicializátory kontextu a jejich přidružené kontextové akceptory.

Tabulka 138. Kontextové iniciátory a jejich přidružené kontextové akceptory

Iniciátor kontextu	**Příjemce kontextu**
MQCHT_CLNTCONN	FUNKCE MQCHT_SVRCONN
PŘÍJEMCE MQCHT_RECEIVER	MQCHT_SENDER
SOUBOR MQCHT_CLURCVR	MQCHT_CLUSDR

Ukončovací program zabezpečení má dva vstupní body:

- **SCSY_NTLM**

  To používá ověřovací služby NTLM, které poskytují jednosměrné ověření. NTLM umožňuje serverům ověřit identity svých klientů. Neumožňuje klientům ověřit identitu serveru nebo jeden server za účelem ověření identity jiného serveru. Ověřování NTLM bylo navrženo pro prostředí sítě, ve kterém se předpokládá, že servery jsou pravé.

- **SCY_KERBEROS**

  To používá vzájemné ověřovací služby Kerberos . Protokol Kerberos nepředpokládá, že servery v síťovém prostředí jsou skutečné. Strany na obou koncích síťového připojení mohou ověřit identitu druhé strany. To znamená, že servery mohou ověřit identitu klientů a jiných serverů a klienti mohou ověřit identitu serveru.

Co dělá uživatelská procedura zabezpečení

Toto téma popisuje, co programy ukončení kanálu SSPI dělají.

Dodané uživatelské programy kanálu poskytují buď jednosměrné, nebo dvoucestné (vzájemné) ověření partnerského systému, když se zavádí relace. Pro konkrétní kanál má každý uživatelský program přidružený *činitel* (podobný ID uživatele, viz "Řízení přístupu IBM MQ a činitelé Windows" na stránce 1000). Spojení mezi dvěma ukončovacími programy je asociací mezi dvěma činiteli.

Po navázání základní relace se ustavuje zabezpečené spojení mezi dvěma programy zabezpečení (jedna pro odesílající agenta MCA a jedna pro přijímajícího agenta MCA). Posloupnost operací je následující:

1. Každý program je přidružen ke konkrétnímu činiteli, například jako výsledek operace explicitního přihlášení.

2. Inicializátor kontextu vyžaduje zabezpečené připojení k partnerovi z balíku zabezpečení (pro Kerberos, jmenovaného partnera) a přijímá token (s názvem token1). Token se odešle s použitím základní relace, která je již zavedena, do partnerského programu.
3. Partnerský program (příjemce kontextu) předává token1 do balíku zabezpečení, který ověřuje, že iniciátor kontextu je autentický. Pro NTLM je nyní navázáno spojení.
4. Pro uživatelskou proceduru zabezpečení dodaný Kerberos(tj. pro vzájemné ověření), vygeneruje balík zabezpečení také druhý token (s názvem token2), který se služba kontextového příjemce vrátí do kontextového iniciátoru pomocí základní relace.
5. Inicializátor kontextu používá token2 k ověření, že je služba Client Acceptor autentická.
6. V této fázi, pokud jsou obě aplikace splněny autentičnosti tokenu partnera, je ustanoveno zabezpečené (ověřené) spojení.

Řízení přístupu IBM MQ a činitelé Windows

Řízení přístupu, které poskytuje produkt IBM MQ , je založeno na uživateli a skupině. Ověřování, které Windows poskytuje, je založeno na činitelích, jako jsou například uživatelé a servicePrincipalName (SPN). V případě názvu servicePrincipalmůže být mnoho z těchto přidružených k jednomu uživateli.

Uživatelská procedura zabezpečení SSPI používá pro ověření příslušné řídicí služby Windows . Je-li ověření Windows úspěšné, uživatelská procedura předá ID uživatele, které je přidružené k činiteli Windows , pro IBM MQ pro řízení přístupu.

Činitelé Windows , kteří jsou relevantní pro ověření, se liší v závislosti na typu použitého ověření.

- Pro ověření NTLM je činitel Windows pro kontext Context Initiator ID uživatele přidruženého k procesu, který je spuštěn. Protože je toto ověření jednosměrné, činitel přidružený k objektu Context acceptor je irelevantní.
- Pro ověření Kerberos na kanálu CLNTCONN je činitel Windows ID uživatele přidružený k procesu, který je spuštěn. Jinak se činitel Windows nachází ve tvaru servicePrincipal, který je vytvořen přidáním následující předpony do názvu QueueManager.

```
ibmMQSeries/
```

z/OS Sestavuje se vaše procedurální aplikace na z/OS

Publikace CICS, IMSa z/OS popisují, jak sestavovat aplikace, které se spouštějí v těchto prostředích.

Tato kolekce témat popisuje další úlohy a změny standardních úloh, které je třeba provést při sestavování aplikací produktu IBM MQ for z/OS pro tato prostředí. Jsou podporovány programovací jazyky COBOL, C, C + +, Assembler a PL/I. (Informace o vytváření aplikací C + + viz Použití C++.)

Úlohy, které musíte provést pro vytvoření spustitelné aplikace IBM MQ for z/OS , závisí na programovacím jazyku, v němž je program napsán, a na prostředí, ve kterém bude aplikace spuštěna.

Kromě kódování volání MQI ve vašem programu přidejte příslušné jazykové příkazy tak, aby zahrnovaly definiční soubor dat produktu IBM MQ for z/OS pro jazyk, který používáte. Seznamte se s obsahem těchto souborů. Úplný popis najdete v tématu “Soubory definic dat produktu IBM MQ” na stránce 686 .

Poznámka

Název **thlqual** je kvalifikátor vyšší úrovně instalační knihovny v systému z/OS.

Příprava programu na spuštění

Poté, co jste napsali program pro aplikaci IBM MQ k vytvoření spustitelné aplikace, musíte jej zkompilovat nebo sestavit a poté propojit výsledný objektový kód s programem stubu, který produkt IBM MQ for z/OS dodává pro každé prostředí, které podporuje.

Jak připravujete svůj program, závisí na prostředí (dávkách, CICS, IMS(BMP nebo MPP), Linux nebo UNIX systémových služeb), v nichž je aplikace spuštěna, a strukturu datových sad na vaší instalaci produktu z/OS .

“Dynamické volání stubu IBM MQ” na stránce 1007 popisuje alternativní metodu vytváření volání MQI ve vašich programech, abyste nepotřebovali propojení-upravit stub stubu IBM MQ . Tato metoda není k dispozici pro všechny jazyky a prostředí.

Nepropojte-upravte vyšší úroveň programu stubu, než je verze produktu IBM MQ for z/OS , na které je váš program spuštěn. Například program spuštěný v produktu MQSeries for OS/390, 5.2 nesmí být upravován odkazem s programem typu stub, který je dodáván s IBM MQ for z/OS 7.

Sestavování 64bitových aplikací v jazyce C
V produktu z/OSjsou aplikace 64bitového C sestaveny pomocí kompilátoru LP64 a voleb vázacího programu. Soubor záhlaví IBM MQ for z/OS *cmqc.h* rozpoznává, je-li tato volba poskytována kompilátoru, a generuje IBM MQ datových typů a struktur, které jsou vhodné pro 64bitovou operaci.

Kód jazyka C vytvořený pomocí této volby musí být sestaven tak, aby používal dynamicky propojené knihovny (DLL) vhodné pro požadovanou sémantickou sémantiku. Při vázání kompilovaného kódu s příslušnou postranní palubou definovanou v části Název postranní plochy vyžadované pro každou koordinační sémantiku se zobrazí specifická knihovna DLL, která je nezbytná.

Tabulka 139. Název podpaluby je vyžadován pro každou sémantickou sémantiku

coordination	**Název postranní plochy**
Rozhraní MQI jednofázového potvrzení	CSQBMQ2X
Dvoufázové potvrzování s koordinací RRS pomocí příkazových slov RRS	CSQBRR2X
Dvoufázové potvrzování s koordinací RRS pomocí příkazových slov MQI	CSQBRI2X

Pomocí procedury EDCQCB JCL, dodávaného s produktem *z/OS XL C/C++*, sestavíte program IBM MQ s jednofázovým potvrzováním jako dávkovou úlohu následujícím způsobem:

```
//PROCS  JCLLIB ORDER=CBC.SCCNPRC
//CLG EXEC EDCQCB,
// INFILE='thlqual.SCSQC37S(CSQ4BCG1)',   < MQ SAMPLES
// CPARM='RENT,SSCOM,DLL,LP64,LIST,NOMAR,NOSEQ',  < COMPILER OPTIONS
// LIBPRFX='CEE',            < PREFIX FOR LIBRARY DSN
// LNGPRFX='CBC',            < PREFIX FOR LANGUAGE DSN
// BPARM='MAP,XREF,RENT,DYNAM=DLL', < LINK EDIT OPTIONS
// OUTFILE='userid.LOAD(CSQ4BCG1),DISP=SHR'
//COMPILE.SYSLIB DD
//        DD
//        DD DISP=SHR,DSN=thlqual.SCSQC370
//BIND.SCSQDEFS DD DISP=SHR,DSN=thlqual.SCSQDEFS
//BIND.SYSIN DD *
INCLUDE SCSQDEFS(CSQBMQ2X)
NAME CSQ4BCG1
```

Chcete-li sestavit koordinovaný program RRS v produktu z/OS Unix System Services, zkompilujte a propojte takto:

```
cc -o mqsamp -W c,LP64,DLL -W l,DYNAM=DLL,LP64 -I"//'thlqual.SCSQC370'" "//'thlqual.SCSQDEFS(CSQBRR2X)'" mqsamp.c
```

z/OS *Sestavení dávkových aplikací produktu z/OS*
Naučte se, jak sestavit dávkové aplikace produktu z/OS a kroky, které je třeba zvážit při jejich provedení.

Chcete-li sestavit aplikaci pro produkt IBM MQ for z/OS spouštěný v rámci dávky produktu z/OS , vytvořte kód JCL (Job Control Language), který provádí tyto úlohy:

1. Kompilace (nebo sestavení) programu k vytvoření objektového kódu. Kód JCL pro vaši kompilaci musí obsahovat příkazy SYSLIB, které zpřístupní soubory definic dat produktu kompilátoru. Definice dat jsou dodávány v následujících knihovnách produktu IBM MQ for z/OS :

- Pro COBOL, **thlqual**.SCSQCOBC
- Pro jazyk assembler, **thlqual**.SCSQMACS
- Pro C, **thlqual**.SCSQC370
- Pro PL/I **thlqual**.SCSQPLIC

2. Pro aplikaci C předpropojte kód objektu vytvořený v kroku “1” na stránce 1001.
3. Pro aplikace PL/I použijte volbu kompilátoru EXTRN (SHORT).
4. Odkaz-editování kódu objektu vytvořeného v kroku “1” na stránce 1001 (nebo kroku “2” na stránce 1002 pro aplikaci C) k vytvoření zaváděcího modulu. Když odkaz upravíte-upravte kód, musíte zahrnout jeden z programů dávkového stubu IBM MQ for z/OS (CSQBSTUB nebo jeden z programů stub RRS: CSQBRRSI nebo CSQBRSTB).

   CSQBSTUB
   jednofázové potvrzování poskytované produktem IBM MQ for z/OS

   CSQBRRSI
   Dvoufázové potvrzování poskytované službou RRS pomocí rozhraní MQI

   CSQBRSTB
   Dvoufázové potvrzování zajišťovaného přímo službou RRS

   Notes:

   a. Používáte-li CSQBRSTB, musíte také propojit svou aplikaci s ATRSCSS z SYS1.CSSLIB. Obrázek 125 na stránce 1002 a Obrázek 126 na stránce 1002 zobrazují fragmenty JCL, aby to bylo možné provést. Stuby jsou jazykově nezávislé a jsou dodávány v knihovně **thlqual**.SCSQLOAD.

   b. Je-li aplikace spuštěna v prostředí jazykového prostředí, měli byste se ujistit, že jste upravovali úpravy pomocí knihovny DLL jazykového prostředí, a nikoli podle pokynů uvedených v tématu “Sestavení dávkových aplikací produktu z/OS pomocí jazykového prostředí” na stránce 1003.

5. Uložte zaváděcí modul do knihovny načtení aplikace.

```
⋮
//*
//* WEBSPHERE MQ FOR Z/OS LIBRARY CONTAINING BATCH STUB
//*
//CSQSTUB   DD  DSN=++THLQUAL++.SCSQLOAD,DISP=SHR
//*
⋮
//SYSIN     DD *
  INCLUDE CSQSTUB(CSQBSTUB)
⋮
/*
```

Obrázek 125. Fragmenty souboru JCL pro propojení-úprava modulu objektu v dávkovém prostředí pomocí jednofázového potvrzení

```
⋮
//*
//* WEBSPHERE MQ FOR Z/OS LIBRARY CONTAINING BATCH STUB
//*
//CSQSTUB   DD  DSN=++THLQUAL++.SCSQLOAD,DISP=SHR
//CSSLIB    DD  DSN=SYS1.CSSLIB,DISP=SHR
//*
⋮
//SYSIN     DD *
INCLUDE CSQSTUB(CSQBRSTB)
INCLUDE CSSLIB(ATRSCSS)
⋮
/*
```

Obrázek 126. Fragmenty souboru JCL pro propojení-úprava modulu objektu v dávkovém prostředí pomocí dvoufázového potvrzování

Chcete-li spustit dávkový nebo RRS program, musíte do zřetězení datové sady STEPLIB nebo JOBLIB zahrnout knihovny **thlqual**.SCSQAUTH a **thlqual**.SCSQLOAD.

Chcete-li spustit program TSO, musíte zahrnout knihovny **thlqual**.SCSQAUTH a **thlqual**.SCSQLOAD v knihovně STEPLIB použité relací TSO.

Chcete-li spustit dávkový program systémových služeb UNIX ze shellu systémových služeb UNIX , přidejte do specifikace STEPLIB ve vašem $HOME..profile knihovny **thlqual**.SCSQAUTH a **thlqual**.SCSQLOAD.

```
STEPLIB= thlqual.SCSQAUTH: thlqual.SCSQLOAD
export STEPLIB
```

z/OS *Sestavení dávkových aplikací produktu z/OS pomocí jazykového prostředí*

Produkt IBM MQ for z/OS poskytuje sadu knihoven DLL (Dynamic Link Library), které musí být použity při linkování vašich aplikací.

Existují dvě varianty knihoven, které aplikaci umožňují použít jedno z následujících volajících rozhraní:

- Rozhraní pro 31bitové rozhraní jazykového prostředí.
- 31bitový volající rozhraní XPLINK. z/OS XPLINK je vysoká konvence volání výkonu dostupná pro aplikace C.

Chcete-li používat knihovny DLL, je aplikace svázána nebo propojena s tzv. *bočnic*namísto stubů dodaných se staršími verzemi. Boční plochy se nacházejí v knihovně SCSQDEFS (místo knihovny SCSQLOAD).

Tabulka 140. Varianty knihoven s dynamickým odkazem

Potvrdit	**31bitové DLL jazykového prostředí**	**31bitová DLL XPLINK**	**Ekvivalentní název stubu**
Knihovny MQI s potvrzením fáze 1	CSQBMQ1	CSQBMQ1X	CSQBSTUB
Dvoufázové potvrzování s koordinováním služby RRS pomocí příkazových příkazových slov RRS	CSQBRR1	CSQBRR1X	CSQBRSTB
Dvoufázové potvrzování s koordinováním služby RRS pomocí příkazových slov MQI transaction-control	CSQBRI1	CSQBRI1X	CSQBRRSI

Poznámka: Všechny bočnice obsahují definici vstupního bodu pro převod dat, MQXCNVC, která byla dříve rozlišena zahrnutím CSQASTUB.

Běžné problémy:

- Následující zpráva se objeví v protokolu úlohy, pokud vaše aplikace používá asynchronní příjem zpráv (volání MQCB, MQCTL nebo MQSUB) a předchozí rozhraní DLL se nepoužívá:

  ```
  CSQB001E Programy jazykového prostředí spuštěné v rámci dávky produktu z/OS nebo USS musí
  používat rozhraní DLL pro produkt IBM MQ .
  ```

  Řešení: Rebuild your application using sidepalub instead of stubs as detailed previously.

- V době sestavení programu se zobrazí následující zpráva:

  ```
  IEW2469E Atributy odkazu na rozhraní MQAPI-NAME z oddílu váš-kód se neshodují s atributy
  Cílový symbol
  ```

  Příčina: To znamená, že jste zkompilovali váš program XPLINK s verzí V701 (nebo novější) verze cmqc.h, ale nejsou vázání s bočnic.

Řešení: Změňte soubor sestavení vašeho programu tak, aby se vázal proti vhodnému podlaží z SCSQDEFS místo stubu z SCSQLOAD

Následující ukázka JCL demonstruje, jak můžete zkompilovat a linkovat program v jazyce C pro použití 31bitového rozhraní jazyka DLL jazykového prostředí:

```
//CLG EXEC EDCCB,
//  INFILE=MYPROGS.CPROGS(MYPROGRAM),
//  CPARM='OPTF(DD:OPTF)',
//  BPARM='XREF,MAP,DYNAM=DLL'           < LINKEDIT OPTIONS
//COMPILE.OPTF DD *
RENT,CHECKOUT(ALL),SSCOM,DEFINE(MVS),NOMARGINS,NOSEQ,DLL
SE(DD:SYSLIBV)
//COMPILE.SYSLIB DD
//               DD
//               DD DISP=SHR,DSN=hlq.SCSQC370
//COMPILE.SYSLIBV DD DISP=SHR,DSN=hlq.BASE.H
/*
//BIND.SYSOBJ DD DISP=SHR,DSN=CEE.SCEEOBJ
//            DD DISP=SHR,DSN=hlq.SCSQDEFS
//BIND.SYSLMOD DD DISP=SHR,DSN=hlq.LOAD(MYPROGAM)
//BIND.SYSIN  DD *
 ENTRY CEESTART
 INCLUDE SYSOBJ(CSQBMQ1)
 NAME MYPROGAM(R)
//
```

Poznámka: Kompilace používá volbu **DLL** . Funkce link-edit používá volbu **DYNAM=DLL** a odkazuje na knihovnu **CSQBMQ1** .

Následující ukázka JCL demonstruje, jak můžete zkompilovat a linkovat program v jazyce C pro použití 31bitového rozhraní XPLINK knihovny DLL:

```
//CLG  EXEC EDCXCB,
//  INFILE=MYPROGS.CPROGS(MYPROGRAM),
//  CPARM='OPTF(DD:OPTF)',
//  BPARM='XREF,MAP,DYNAM=DLL'           < LINKEDIT OPTIONS
//COMPILE.OPTF DD *
RENT,CHECKOUT(ALL),SSCOM,DEFINE(MVS),NOMARGINS,NOSEQ,XPLINK,DLL
SE(DD:SYSLIBV)
//COMPILE.SYSLIB DD
//               DD
//               DD DISP=SHR,DSN=hlq.SCSQC370
//COMPILE.SYSLIBV DD DISP=SHR,DSN=hlq.BASE.H
/*
//BIND.SYSOBJ DD DISP=SHR,DSN=CEE.SCEEOBJ
//            DD DISP=SHR,DSN=hlq.SCSQDEFS
//BIND.SYSLMOD DD DISP=SHR,DSN=hlq.LOAD(MYPROGAM)
//BIND.SYSIN  DD *
 ENTRY CEESTART
 INCLUDE SYSOBJ(CSQBMQ1X)
 NAME MYPROGAM(R)
//
```

Poznámka: Kompilace používá volby **XPLINK** a **DLL** . Odkaz na úpravu používá volbu **DYNAM=DLL** a odkazuje na knihovnu **CSQBMQ1X** .

Ujistěte se, že jste do každého programu v modulu přidali knihovnu DLL pro kompilaci všech programů. Zprávy jako IEW2456E 9207 SYMBOL CSQ1BAK NEVYŘEŠENO označuje, že je třeba zkontrolovat, zda byly všechny programy zkompilovány s volbou DLL.

Sestavování aplikací produktu CICS v produktu z/OS
Tyto informace použijte při sestavování aplikací produktu CICS v produktu z/OS.

Chcete-li sestavit aplikaci pro produkt IBM MQ for z/OS , který běží pod CICS, musíte:

- Přeložte příkazy CICS ve vašem programu do jazyka, ve kterém je zapisován zbytek vašeho programu.
- Kompilujte nebo sestavte výstup z překladače a vytvořte kód objektu.
  - V případě programů PL/I použijte volbu kompilátoru EXTRN (SHORT).

  - Pro aplikace v jazyku C platí, že pokud aplikace nepoužívá příkaz XPLINK, použijte volbu kompilátoru DEFINE (MQ_OS_LINKAGE=1).
- Odkaz-upravte kód objektu za účelem vytvoření zaváděcího modulu.

CICS poskytuje proceduru pro provedení těchto kroků v posloupnosti pro každý programovací jazyk, který podporuje.

- Pro CICS Transaction Server for z/OS, příručka *CICS Transaction Server for z/OS System Definition Guide* popisuje použití těchto procedur a příručky *CICS/ESA Application Programming Guide* poskytuje více informací o procesu překladu.

Musíte zahrnout:

- V příkazu SYSLIB kompilační (nebo montážní) fáze, příkazy, které činí soubory definic dat produktu dostupné kompilátoru. Definice dat jsou dodávány v následujících knihovnách produktu IBM MQ for z/OS :
  - Pro COBOL, **thlqual**.SCSQCOBC
  - Pro jazyk assembler, **thlqual**.SCSQMACS
  - Pro C, **thlqual**.SCSQC370
  - Pro PL/I **thlqual**.SCSQPLIC
- Ve svém kódu JCL linkování program stubu produktu IBM MQ for z/OS CICS (CSQCSTUB). Obrázek 127 na stránce 1005 zobrazuje fragmenty kódu JCL, aby to bylo možné provést. Stub je závislý na jazyku a je dodán v knihovně **thlqual**.SCSQLOAD.

```
⋮
//*
//* WEBSPHERE MQ FOR Z/OS LIBRARY CONTAINING CICS STUB
//*
//CSQSTUB   DD  DSN=++THLQUAL++.SCSQLOAD,DISP=SHR
//*
⋮
//LKED.SYSIN DD *
  INCLUDE CSQSTUB(CSQCSTUB)
  ⋮
/*
```

Obrázek 127. Fragmenty souboru JCL pro propojení-úprava modulu objektu v prostředí produktu CICS

- V případě produktu CICS verze novější než CICS TS 3.2nebo pokud chcete použít rozhraní API vlastností zpráv produktu IBM MQ nebo rozhraní API IBM MQ MQCB, MQCTL, MQSTAT, MQSUB nebo MQSUBR, musíte upravit kód objektu pomocí stubu dodaného s produktem CICS , DFHMQSTB a nikoli produktu IBM MQ CSQCSTUB. Další informace o sestavování programů produktu IBM MQ pro produkt CICSnajdete v tématu Program stub rozhraní API pro přístup k voláním produktu IBM MQ MQI v dokumentaci produktu CICS .

Po provedení těchto kroků uložte zaváděcí modul do knihovny načtení aplikace a definujte program obvyklým způsobem v produktu CICS .

Before you run a CICS program, your system administrator must define it to CICS as an IBM MQ program and transaction, You can then run it in the typical way.

Sestavování aplikací IMS (BMP nebo MPP)
Tyto informace použijte při sestavování aplikací IMS (BMP nebo MPP).

Pokud sestavujete dávkové programy DL/I, prostudujte si téma “Sestavení dávkových aplikací produktu z/OS” na stránce 1001. Chcete-li sestavit jiné aplikace, které jsou spuštěny v produktu IMS (jako BMP nebo MPP), vytvořte JCL, které tyto úlohy provádí:

1. Kompilace (nebo sestavení) programu k vytvoření objektového kódu. Kód JCL pro vaši kompilaci musí obsahovat příkazy SYSLIB, které zpřístupní soubory definic dat produktu kompilátoru. Definice dat jsou dodávány v následujících knihovnách produktu IBM MQ for z/OS :
   - Pro COBOL, **thlqual**.SCSQCOBC

- Pro jazyk assembler, **thlqual**.SCSQMACS
- Pro C, **thlqual**.SCSQC370
- Pro PL/I, **thlqual**.SCSQPLIC

2. Pro aplikaci C propojte modul objektu vytvořený v kroku “1” na stránce 1005.
3. V případě programů PL/I použijte volbu kompilátoru EXTRN (SHORT).
4. Pokud aplikace nepoužívá aplikaci XPLINK, v případě aplikace C použijte volbu kompilátoru DEFINE (MQ_OS_LINKAGE=1).
5. Odkaz-editování kódu objektu vytvořeného v kroku “1” na stránce 1005 (nebo kroku “2” na stránce 1006 pro aplikaci C/370) k vytvoření zaváděcího modulu:
   a. Zahrňte modul rozhraní jazyka produktu IMS (DFSLI000).
   b. Zahrnout program stubu IBM MQ for z/OS IMS (CSQQSTUB). Obrázek 128 na stránce 1006 zobrazuje fragmenty JCL, aby to bylo možné provést. Stub je nezávislý na jazyku a je dodán v knihovně **thlqual**.SCSQLOAD.

      Poznámka: Používáte-li COBOL, vyberte volbu kompilátoru NODYNAM, chcete-li umožnit, aby editor sestavení interpretoval odkazy na CSQQSTUB, pokud nechcete použít dynamické propojení, jak je popsáno v tématu “Dynamické volání stubu IBM MQ” na stránce 1007.
6. Uložte zaváděcí modul do knihovny načtení aplikace.

```
⋮
//*
//* WEBSPHERE MQ FOR Z/OS LIBRARY CONTAINING IMS STUB
//*
//CSQSTUB   DD  DSN=thlqual.SCSQLOAD,DISP=SHR
//*
⋮
//LKED.SYSIN DD *
  INCLUDE CSQSTUB(CSQQSTUB)
  ⋮
/*
```

Obrázek 128. Fragmenty souboru JCL pro propojení-úprava modulu objektu v prostředí produktu IMS

Before you run an IMS program, your system administrator must define it to IMS as an IBM MQ program and transaction: you can then run it in the typical way.

Sestavování aplikací produktu z/OS UNIX System Services
Tyto informace použijte při sestavování aplikací produktu z/OS UNIX System Services.

Chcete-li sestavit aplikaci jazyka C pro produkt IBM MQ for z/OS , který běží pod systémem UNIX System Services, zkompilujte a propojte svou aplikaci takto:

```
cc -o mqsamp -W c,DLL -I "//' thlqual.SCSQC370'" mqsamp.c "//' thlqual.SCSQDEFS(CSQBMQ1)'"
```

kde **thlqual** je kvalifikátor vyšší úrovně, který jste použili při instalaci.

Chcete-li spustit program v jazyce C, musíte do svého souboru `.profile` přidat následující text; mělo by to být ve vašem kořenovém adresáři:

```
STEPLIB= thlqual.SCSQANLE:thlqual.SCSQAUTH: STEPLIB
```

Všimněte si, že je třeba ukončit produkt UNIX System Services a znovu zadat UNIX Systémové služby, aby byla změna rozeznána.

Chcete-li spustit více shellů, přidejte na začátek řádku export slov, tj.:

```
export STEPLIB= thlqual.SCSQANLE:thlqual.SCSQAUTH: STEPLIB
```

Po úspěšném dokončení můžete spojit CSQBSTUB a vydávat volání IBM MQ .

"Dynamické volání stubu IBM MQ" na stránce 1007 popisuje alternativní metodu vytváření volání MQI ve vašich programech, abyste nepotřebovali propojení-upravit stub stubu IBM MQ . Tato metoda není k dispozici pro všechny jazyky a prostředí.

Nepropojte-upravte vyšší úroveň programu stubu, než je verze produktu IBM MQ for z/OS , na které je váš program spuštěn. Například program spuštěný v produktu IBM WebSphere MQ for z/OS 7.1 nesmí být upravován pomocí propojení s programem stubu dodávaným s produktem IBM MQ for z/OS 8.0.

Dynamické volání stubu IBM MQ

Místo odkazu-úprava programu stubu IBM MQ s vaším kódem objektu, můžete dynamicky volat stub z vašeho programu.

To lze provést v prostředí dávkového zpracování, produktu IMSa CICS . Tato funkce není podporována v prostředí RRS. Pokud váš aplikační program používá služby RRS ke koordinaci aktualizací, prostudujte si téma "Podmínky RRS" na stránce 1011.

Nicméně tato metoda:

- Zvyšuje složitost vašich programů.
- Zvýší paměť požadovanou pro programy v době provádění
- Omezuje výkon vašich programů.
- Znamená to, že nemůžete používat stejné programy v jiných prostředích.

Pokud se stub volá dynamicky, musí být v době provádění k dispozici příslušný spojkový program a jeho aliasy. Chcete-li to zajistit, zahrňte datovou sadu IBM MQ for z/OS SCSQLOAD:

- Pro dávkové zpracování a IMSve zřetězení STEPLIB souboru JCL.
- Pro CICS, v zřetězení CICS DFHRPL.

U produktu IMSse ujistěte, že knihovna obsahující dynamický stub (built as described in the information about installing the IMS adapter in Setting up the IMS adapter) je napřed před datovou sadou SCSQLOAD ve zřetězení STEPLIB v oblasti JCL oblasti.

Použijte názvy zobrazené v Tabulka 141 na stránce 1007 při dynamickém volání stubu. V jazyku PL/I deklarujte pouze názvy volání použité ve vašem programu.

Tabulka 141. Názvy volání pro dynamické propojení

Volání rozhraní MQI	Názvy dynamických volání dávkových úloh (mimo RRS)	Dynamické názvy volání produktu CICS	Dynamické názvy volání produktu IMS
MQBACK	CSQBBACK	není podporováno	Nepodporováno
MQBUFMH5	CSQBBFMH	CSQCBFMH [1]	MQBUFMH5
MQCB	CSQBCB	CSQCCB [1]	Nepodporováno
MQCLOSE	CSQBCLOS	CSQCCLOS	MQCLOSE
MQCMIT	CSQBCOMM	není podporováno	Nepodporováno
MQCONN	CSQBCONN	PŘIPOJENÍ CSQCCONN	MQCONN
MQCONNX	CSQBCONX	CSQCCONX	MQCONNX
MQCRTM	CSQBCTMH	CSQCCTMH [1]	MQCRTM
MQCTL	CSQBCTL	CSQCCTL [1]	Nepodporováno
MQDISC	CSQBDISC	CSQCDISC	MQDISC
MQDLTMH	CSQBTMH	CSQCDTMH [1]	MQDLTMH
MQDLTMP	CSQBDTMP	CSQCDTMP [1]	MQDLTMP

Tabulka 141. Názvy volání pro dynamické propojení (pokračování)			
Volání rozhraní MQI	**Názvy dynamických volání dávkových úloh (mimo RRS)**	**Dynamické názvy volání produktu CICS**	**Dynamické názvy volání produktu IMS**
MQGET	CSQBGET	CSQCGET	MQGET
MQINQ	CSQBINQ	CSQCINQ	MQINQ
MQINQMP	CSQBIQMP	CSQCIQMP [1]	MQINQMP
MQMHBUF	CSQBMHBF	CSQCMHBF [1]	MQMBUF
MQOPEN	CSQBOPEN	CSQCOPEN	MQOPEN
MQPUT	CSQBPUT	CSQCPUT	MQPUT
MQPUT1	CSQBPUT1	CSQCPUT1	MQPUT1
MQSET	CSQBSET	CSQCSET	MQSET
MQSETMP	CSQBSTMP	CSQCSTMP [1]	MQSETMP
Příkaz MQSTAT	CSQBSTAT	CSQCSTAT [1]	MQSTAT
MQSUB	CSQBSUB	CSQCSUB [1]	MQSUB
MQSUBRQ.	CSQBSUBR	CSQCSUBR [1]	MQSUBRQ.

Poznámka: 1. Tato volání rozhraní API jsou k dispozici pouze při použití CICS TS 3.2 nebo novější a musí být použit CSQCSTUB dodávaný s CICS . Pro produkt CICS TS 3.2musí být použita oprava APAR PK66866 . Pro produkt CICS TS 4.1musí být použita oprava APAR PK89844 .

Příklady použití této techniky naleznete v následujících obrázcích:

- Dávka a COBOL: viz Obrázek 129 na stránce 1008
- CICS a COBOL: viz Obrázek 130 na stránce 1009
- IMS a COBOL: viz Obrázek 131 na stránce 1009
- Dávková a assembler: viz Obrázek 132 na stránce 1009
- CICS a assembler: viz Obrázek 133 na stránce 1009
- IMS a assembler: viz Obrázek 134 na stránce 1010
- Dávka a C: Obrázek 135 na stránce 1010
- CICS a C: viz Obrázek 136 na stránce 1010
- IMS a C: viz Obrázek 137 na stránce 1010
- Dávka a PL/I: viz Obrázek 138 na stránce 1010
- IMS a PL/I: viz Obrázek 139 na stránce 1011

```
...
      WORKING-STORAGE SECTION.
...
         05 WS-MQOPEN                    PIC X(8) VALUE 'CSQBOPEN'.
...
      PROCEDURE DIVISION.
...
          CALL WS-MQOPEN WS-HCONN
                         MQOD
                         WS-OPTIONS
                         WS-HOBJ
                         WS-COMPCODE
                         WS-REASON.
...
```

Obrázek 129. Dynamické propojování pomocí jazyka COBOL v dávkovém prostředí

```
...
       WORKING-STORAGE SECTION.
...
          05 WS-MQOPEN                     PIC X(8) VALUE 'CSQCOPEN'.
...
       PROCEDURE DIVISION.
...
           CALL WS-MQOPEN WS-HCONN
                          MQOD
                          WS-OPTIONS
                          WS-HOBJ
                          WS-COMPCODE
                          WS-REASON.
...
```

Obrázek 130. Dynamické propojování pomocí jazyka COBOL v prostředí produktu CICS

```
...
       WORKING-STORAGE SECTION.
...
          05 WS-MQOPEN                     PIC X(8) VALUE 'MQOPEN'.
...
       PROCEDURE DIVISION.
...
           CALL WS-MQOPEN WS-HCONN
                          MQOD
                          WS-OPTIONS
                          WS-HOBJ
                          WS-COMPCODE
                          WS-REASON.
...
      * ------------------------------------------------------------ *
      *
      *    If the compilation option 'DYNAM' is specified
      *    then you may code the MQ calls as follows
      *
      * ------------------------------------------------------------ *
...
           CALL 'MQOPEN'  WS-HCONN
                          MQOD
                          WS-OPTIONS
                          WS-HOBJ
                          WS-COMPCODE
                          WS-REASON.
...
```

Obrázek 131. Dynamické propojování pomocí jazyka COBOL v prostředí produktu IMS

```
...
         LOAD   EP=CSQBOPEN
...
         CALL (15),(HCONN,MQOD,OPTIONS,HOBJ,COMPCODE,REASON),VL
...
         DELETE EP=CSQBOPEN
...
```

Obrázek 132. Dynamické propojování s použitím jazyka sestavení v dávkovém prostředí

```
...
         EXEC CICS LOAD PROGRAM('CSQCOPEN') ENTRY(R15)
...
         CALL (15),(HCONN,MQOD,OPTIONS,HOBJ,COMPCODE,REASON),VL
...
         EXEC CICS RELEASE PROGRAM('CSQCOPEN')
...
```

Obrázek 133. Dynamické propojování s použitím jazyka sestavení v prostředí produktu CICS

```
...
         LOAD   EP=MQOPEN
...
         CALL (15),(HCONN,MQOD,OPTIONS,HOBJ,COMPCODE,REASON),VL
...
         DELETE EP=MQOPEN
...
```

Obrázek 134. Dynamické propojování s použitím jazyka sestavení v prostředí produktu IMS

```
...
typedef void CALL_ME();
#pragma linkage(CALL_ME, OS)
...
main()
{
CALL_ME * csqbopen;
...
csqbopen = (CALL_ME *) fetch("CSQBOPEN");
(*csqbopen)(Hconn,&ObjDesc,Options,&Hobj,&CompCode,&Reason);
...
```

Obrázek 135. Dynamické propojování pomocí jazyka C v prostředí dávkového zpracování

```
...
typedef void CALL_ME();
#pragma linkage(CALL_ME, OS)
...
main()
{
CALL_ME * csqcopen;
...
  EXEC CICS LOAD PROGRAM("CSQCOPEN") ENTRY(csqcopen);
(*csqcopen)(Hconn,&ObjDesc,Options,&Hobj,&CompCode,&Reason);
...
```

Obrázek 136. Dynamické propojování pomocí jazyka C v prostředí CICS

```
...
typedef void CALL_ME();
#pragma linkage(CALL_ME, OS)
...
main()
{
CALL_ME * mqopen;
...
mqopen = (CALL_ME *) fetch("MQOPEN");
(*mqopen)(Hconn,&ObjDesc,Options,&Hobj,&CompCode,&Reason);
...
```

Obrázek 137. Dynamické propojování pomocí jazyka C v prostředí IMS

```
...
      DCL CSQBOPEN ENTRY EXT OPTIONS(ASSEMBLER INTER);
...
      FETCH CSQBOPEN;

      CALL CSQBOPEN(HQM,
                    MQOD,
                    OPTIONS,
                    HOBJ,
                    COMPCODE,
                    REASON);

      RELEASE CSQBOPEN;
```

Obrázek 138. Dynamické propojování pomocí jazyka PL/I v dávkovém prostředí

```
...
      DCL MQOPEN   ENTRY EXT OPTIONS(ASSEMBLER INTER);
...
      FETCH MQOPEN;

      CALL   MQOPEN(HQM,
                    MQOD,
                    OPTIONS,
                    HOBJ,
                    COMPCODE,
                    REASON);

      RELEASE   MQOPEN;
```

Obrázek 139. Dynamické propojování pomocí jazyka PL/I v prostředí produktu IMS

Podmínky RRS
Zvažte použití těchto informací, pokud váš aplikační program používá služby RRS ke koordinaci aktualizací.

Produkt IBM MQ poskytuje dva různé stuby pro dávkové programy, které potřebují koordinaci RRS- viz "Dávkový adaptér RRS" na stránce 854. Rozdíl v chování pozdějších volání API je určen v době MQCONN adaptérem dávky z informací předaných rutinou stubu na rozhraní MQCONN nebo MQCONNX API. To znamená, že volání dynamických rozhraní API jsou k dispozici pro dávkové programy, které potřebují koordinaci služby RRS, za předpokladu, že počáteční připojení k produktu IBM MQ bylo provedeno pomocí příslušného stubu. Následující příklad ilustruje následující:

```
       WORKING-STORAGE SECTION.
           05 WS-MQOPEN       PIC X(8) VALUE 'MQOPEN' .
.
.
.
       PROCEDURE DIVISION.
.
.
.
       *
       * Static call to MQCONN must be resolved by linkage edit to
       * CSQBRSTB or CSQBRRSI for RRS coordination
       *
           CALL 'MQCONN' USING W00-QMGR
                               W03-HCONN
                               W03-COMPCODE
                               W03-REASON.
.
.
.
       *
           CALL WS-MQOPEN  WS-HCONN
                           MQOD
                           WS-OPTIONS
                           WS-HOBJ
                           WS-COMPCODE
                           WS-REASON.
```

Ladění vašich programů

Tyto informace použijte k seznámení se s ladicími programy TSO a CICS a s prozírami do trasování produktu CICS .

Hlavní pomocné programy pro ladění aplikačních programů IBM MQ for z/OS jsou kódy příčiny, které jsou vraceny každým voláním rozhraní API. Seznam těchto akcí včetně nápadů pro nápravnou akci naleznete v následujících tématech:

- položky Zprávy, dokončení a kódy příčiny produktu IBM MQ for z/OS pro IBM MQ for z/OS
- Zprávy a kódy příčin pro všechny ostatní platformy IBM MQ

Toto téma také navrhuje další nástroje ladění, které je třeba použít v konkrétních prostředích.

Ladění programů TSO

Pro programy TSO jsou k dispozici následující interaktivní nástroje ladění:

- Nástroj TEST
- VS interaktivní ladicí nástroj COBOL II
- ZKONTROLOVAT interaktivní nástroj ladění pro programy v jazycích C a PL/I

Ladění programů programu CICS

Diagnostický nástroj provedení CICS (CEDF) můžete použít k interaktivnímu testování programů produktu CICS bez nutnosti úpravy programu nebo procedury přípravy programu.

Další informace o EDF naleznete v příručce *CICS Transaction Server for z/OS CICS Application Programming Guide*.

CICS trasování

Pravděpodobně také zjistíte, že je užitečné použít CICS Trasovací řídicí transakci (CETR) k řízení aktivity trasování produktu CICS .

Další informace o CETR naleznete v příručce *CICS Transaction Server for z/OS CICS-Supplied Transactions* .

Chcete-li určit, zda je trasování CICS aktivní, zobrazte stav připojení pomocí panelu CKQC. Tento panel také zobrazuje trasovací číslo.

Chcete-li interpretovat trasovací záznamy CICS , prohlédněte si téma Tabulka 142 na stránce 1012.

Trasovací položka CICS pro tyto hodnoty je AP0 *xxx* (kde *xxx* je číslo trasování zadané při povolení adaptéru CICS). Všechny trasovací záznamy kromě CSQCTEST jsou vydávány CSQCTRUE. CSQCTEST je vydán CSQCRST a CSQCDSP.

Tabulka 142. Trasovací položky adaptéru CICS

Název	Popis	Posloupnost trasování	Trasovací data
CSQCABNT	Nestandardní ukončení	Před zadáním parametru END_THREAD ABNORMAL do produktu IBM MQ. Důvodem je ukončení úlohy a implicitní vrácení může být provedeno aplikací. Požadavek ROLLBACK je zahrnut do volání END_THREAD v tomto případě.	Informace o jednotce práce. Tyto informace můžete použít při zjišťování o stavu práce. (Může být například ověřeno proti výstupu vytvořenému příkazem DISPLAY THREAD nebo s obslužným programem pro tisk protokolu produktu IBM MQ for z/OS .)
CSQCBACK	Synchronizační bod Syncpoint	Před zadáním BACKOUT do IBM MQ for z/OS. Důvodem je explicitní žádost o odvolání z aplikace.	Informace o jednotce práce.
CSQCCCRC	Kód dokončení a kód příčiny	Po neúspěšném návratu z volání API.	Kód dokončení a kód příčiny.

Tabulka 142. Trasovací položky adaptéru CICS (pokračování)

Název	Popis	Posloupnost trasování	Trasovací data
CSQCCOMM	Potvrzení synchronizačního bodu	Před vydáním příkazu COMMIT pro IBM MQ for z/OS. Důvodem může být požadavek jednofázového potvrzení nebo druhá fáze dvoufázového požadavku na potvrzení. Požadavek je způsoben explicitním požadavkem synchronizačního bodu z aplikace.	Informace o jednotce práce.
CSQCEXER	Provedení vyřešení	Před vydáním EXECUTE_RESOLVE na IBM MQ for z/OS.	Informaci o pracovní jednotce jednotky práce, která vydala EXECUTE_RESOLVE. Jedná se o poslední neověřenou jednotku práce v procesu resynchronizace.
CSQCGETW.	Čekání GET	Před vydáním příkazu CICS wait.	Adresa ECB, na kterou se bude čekat.
CSQGGDD	Data zprávy GET	Po úspěšném návratu z příkazu MQGET.	Až 40 bajtů dat zprávy.
CSQGGGH	Popisovač zprávy GET	Před zadáním příkazu MQGET do produktu IBM MQ for z/OS.	Popisovač objektu.
CSQCGMGI	Získat ID zprávy	Po úspěšném návratu z příkazu MQGET.	ID zprávy a ID korelace zprávy.
CSQCINDL	Seznam nejistých položek	Po úspěšném návratu z druhého INQUIRE_INDOUBT.	Neověřené jednotky seznamu prací.
CSQCINDO	Pouze IBM		
CSQCINDS	Velikost seznamu neověřených položek	Po úspěšném návratu z prvního INQUIRE_INDOUBT a neověřený seznam není prázdný.	Délka seznamu. Dělení do 64 dává počet neověřených jednotek práce.
CSQCINQH	Popisovač INQ	Před vydáním MQINQ do IBM MQ for z/OS.	Popisovač objektu.
CSQCLOSH	Popisovač CLOSE	Před vydáním příkazu MQCLOSE do produktu IBM MQ for z/OS.	Popisovač objektu.
CSQCLOST	Odebrání bylo ztraceno	Během procesu resynchronizace produkt CICS informuje adaptér, že byl restartován, takže nejsou k dispozici žádné informace o odebrání jednotky práce, která má být znovu synchronizována.	ID pracovní jednotky známé produktu CICS pro jednotku práce, která má být znovu synchronizována.
CSQCNIND	Umístění není neověřené	Během procesu resynchronizace produkt CICS informuje adaptér o tom, že jednotka práce, která se má znovu synchronizovat, by neměla být neověřena (to znamená, že je možná stále spuštěna).	ID pracovní jednotky známé produktu CICS pro jednotku práce, která má být znovu synchronizována.

Tabulka 142. Trasovací položky adaptéru CICS (pokračování)

Název	Popis	Posloupnost trasování	Trasovací data
CSQCNORT	Běžné ukončení	Před zadáním parametru END_THREAD NORMAL do produktu IBM MQ for z/OS. Důvodem je konec úlohy, a proto může aplikace provést implicitní potvrzení synchronizačního bodu. Požadavek COMMIT je součástí volání END_THREAD v tomto případě.	Informace o jednotce práce.
CSQCOPNH	Manipulátor OPEN	Po úspěšném návratu z MQOPEN.	Popisovač objektu.
CSQCOPNO	Objekt OPEN	Před vydáním MQOPEN do IBM MQ for z/OS.	Název objektu.
CSQCPMGD	Data zprávy PUT	Před vydáním MQPUT do IBM MQ for z/OS.	Až 40 bajtů dat zprávy.
CSQCPMGH	Popisovač zprávy PUT	Před vydáním MQPUT do IBM MQ for z/OS.	Popisovač objektu.
CSQCPMGI	ID zprávy PUT	Po úspěšném volání MQPUT z produktu IBM MQ for z/OS.	ID zprávy a ID korelace zprávy.
CSQCPREP	Příprava synchronizačního bodu	Před vydáním příkazu PREPARE k produktu IBM MQ for z/OS v první fázi zpracování dvoufázového potvrzování. Toto volání může být také vydáno z komponenty distribuované fronty jako volání API.	Informace o jednotce práce.
CSQCP1MD	Data zprávy PUTONE	Před vydáním příkazu MQPUT1 do produktu IBM MQ for z/OS.	Až 40 bajtů dat zprávy.
CSQCP1MI	ID zprávy PUTONE	Po úspěšném návratu z MQPUT1.	ID zprávy a ID korelace zprávy.
CSQCP1ON	Název objektu PUTONE	Před vydáním příkazu MQPUT1 do produktu IBM MQ for z/OS.	Název objektu.
CSQCRBAK	Vyřešené odvolání	Před vydáním RESOLVE_ROLLBACK na IBM MQ for z/OS.	Informace o jednotce práce.
CSQCRCMT	Vyhodnocené potvrzení	Před vydáním RESOLVE_COMMIT na IBM MQ for z/OS.	Informace o jednotce práce.

Tabulka 142. Trasovací položky adaptéru CICS (pokračování)			
Název	**Popis**	**Posloupnost trasování**	**Trasovací data**
CSQCRMIR	Odezva RMI	Před návratem k vyvolání RMI (Resource manager) produktu CICS z určitého vyvolání.	Architovaná hodnota odpovědi RMI. Jeho význam závisí na typu vyvolání. Tyto hodnoty jsou dokumentovány v příručce *CICS Transaction Server for z/OS Customization Guide*. Chcete-li určit typ vyvolání, podívejte se na předchozí trasovací záznamy vytvořené komponentou vyvolání RMI produktu CICS .
CSQCRCYN	Resynchronizace	Před spuštěním procesu resynchronizace pro úlohu.	ID pracovní jednotky známé produktu CICS pro jednotku práce, která má být znovu synchronizována.
CSQCSETH	popisovač SET	Před zadáním příkazu MQSET do produktu IBM MQ for z/OS.	Popisovač objektu.
CSQCTASA	Pouze IBM		
CSQCTEST	Test trasování	Používá se ve volání EXEC CICS ENTER TRACE k ověření trasovacího čísla zadaného uživatelem nebo stavem trasování připojení.	Žádná data.
CSQCDCFF	Pouze IBM		

Obsluha chyb procedurálních programů

Tyto informace vysvětlují chyby související s voláními MQI MQI buď při volání, nebo při doručení její zprávy do konečného cíle.

Kdykoli je to možné, vrátí správce front veškeré chyby, jakmile je provedeno volání MQI. Jedná se o *lokálně zjištěné chyby*.

Při odesílání zpráv do vzdálené fronty nemusí být při volání MQI zjevné chyby. V takovém případě správce front, který identifikuje chyby, ohlásí odeslání jiné zprávy do původního programu. Jedná se o *vzdáleně určené chyby*.

Lokálně určené chyby

Informace o lokálně určených chybách, které zahrnují: selhání při volání MQI, přerušení systému a zprávy obsahující nesprávná data.

Tři nejčastější příčiny chyb, které správce front může hlásit okamžitě, jsou:

- Selhání volání MQI; například, protože je plná fronta
- Přerušení běhu některé části systému, na které závisí vaše aplikace; například správce front.
- Zprávy obsahující data, která nelze úspěšně zpracovat

Používáte-li asynchronní prostředek vložení, chyby se neohlašují okamžitě. Použijte volání MQSTAT k načtení informací o stavu pro předchozí asynchronní operace put.

Selhání volání MQI

Správce front může okamžitě nahlásit případné chyby v kódování volání MQI. To se provádí pomocí souboru předdefinovaných návratových kódů. Ty jsou rozděleny do kódů dokončení a kódů příčiny.

Chcete-li zobrazit, zda je volání úspěšné, správce front vrátí při dokončení volání *kód dokončení* . K dispozici jsou tři kódy dokončení označující úspěch, částečné dokončení a selhání volání. Správce front také vrátí *kód příčiny* , který indikuje důvod částečného dokončení nebo selhání volání.

Kódy dokončení a příčiny pro každé volání jsou vypsány s popisem daného volání v tématu Návratové kódy. Podrobnější informace, včetně nápadů pro nápravnou akci, najdete v tématu:

- z/OS položky Zprávy, dokončení a kódy příčiny produktu IBM MQ for z/OS pro IBM MQ for z/OS
- Zprávy a kódy příčin pro všechny ostatní platformy IBM MQ

Navrhněte své programy ke zpracování všech návratových kódů, které mohou nastat při každém hovoru.

System initrorupcí

Pokud správce front, k němuž je připojen, se musí zotavit ze selhání systému, může být vaše aplikace nevědomá žádné přerušení. Je však třeba navrhnout aplikaci tak, aby nedošlo ke ztrátě dat, dojde-li k takovému přerušení.

Metody, které můžete použít k zajištění konzistence dat, závisí na platformě, na které je správce front spuštěn:

z/OS z/OS
V prostředí CICS a IMS můžete provést volání MQPUT a MQGET v rámci jednotek práce, které jsou spravovány produktem CICS nebo IMS. V prostředí dávkového zpracování můžete provést volání MQPUT a MQGET stejným způsobem, ale je nutné deklarovat synchronizační body pomocí následujících kroků:

- Volání IBM MQ for z/OS MQCMIT a MQBACK (viz “Potvrzení a zálohování jednotek práce” na stránce 818), nebo
- Služby z/OS Transaction Management a Recoverable Resource Manager Services (RRS) poskytují podporu dvoufázového synchronizačního bodu. Služba RRS vám umožňuje aktualizovat produkt IBM MQ i další prostředky produktu s podporou služby RRS, jako jsou prostředky uložené procedury produktu Db2 , v rámci jedné logické pracovní jednotky. Informace o podpoře synchronizačních bodů RRS viz “Správa transakcí a zotavitelné služby správce prostředků” na stránce 822.

IBM i IBM i
Volání MQPUT a MQGET můžete provést v rámci globálních transakcí, které jsou spravovány vázaným zpracováním IBM i . Synchronizační body můžete deklarovat pomocí nativních příkazů IBM i COMMIT a ROLLBACK nebo pomocí příkazů specifických pro daný jazyk. Lokální jednotky práce jsou spravovány produktem IBM MQ pomocí volání MQCMIT a MQBACK.

Systémy UNIX, Linux, and Windows
V těchto prostředích můžete provést volání MQPUT a MQGET obvyklým způsobem, je však nutné deklarovat synchronizační body pomocí volání MQCMIT a MQBACK (viz “Potvrzení a zálohování jednotek práce” na stránce 818). V prostředí produktu CICS jsou příkazy MQCMIT a MQBACK zakázány, protože můžete provést volání MQPUT a MQGET v rámci jednotek práce, které jsou spravovány produktem CICS.

Použít trvalé zprávy pro provedení všech dat, které si nemůžete dovolit ztratit. Trvalé zprávy jsou obnoveny ve frontách, pokud se správce front musí zotavit ze selhání. ULW S produktem IBM MQ v systému UNIX, Linux, and Windowsdojde k selhání volání MQGET nebo MQPUT v rámci vaší aplikace při zaplnění všech souborů protokolu se zprávou MQRC_RESOURCE_PROBLÉM. Další informace o souborech protokolu v systému UNIX, Linux, and Windowsviz Správa. z/OS Pro z/OS viz Plánování v systému z/OS.

Pokud je správce front v době spuštění aplikace zastaven operátorem, je obvykle použita volba uvedení do klidového stavu. Správce front přejde do klidového stavu, ve kterém mohou aplikace pokračovat v práci, ale musí být ukončeny co nejdříve. Malé a rychlé aplikace mohou pravděpodobně ignorovat stav uvedení do klidového stavu a pokračovat, dokud nebudou ukončeny normální. Delší běžící aplikace nebo ty, které čekají na příchod zpráv, by měly použít volbu *selže při uvedení do klidového stavu* , když používají volání MQOPEN, MQPUT, MQPUT1a MQGET. Tyto volby znamenají, že volání selže, když správce front přechází do klidového stavu, ale aplikace může mít stále čas k tomu, aby mohla být ukončena čistě, vydáním volání, která ignorují stav uvedení do klidového stavu. Takové aplikace by mohly také potvrdit nebo vrátit zpět změny, které provedli, a poté ukončit.

Je-li správce front přinucen zastavit (tj. ukončit bez uvedení do klidového stavu), aplikace obdrží při volání MQI příkaz MQRC_CONNECTION_BROKEN kódu příčiny. Ukončete aplikaci nebo případně na systémech IBM i IBM MQ for IBM i, UNIX, Linux, and Windows zadejte volání MQDISC.

Zprávy obsahující nesprávná data

Použijete-li jednotky práce ve vaší aplikaci, pokud program nemůže úspěšně zpracovat zprávu, kterou načítá z fronty, je volání MQGET vráceny.

Správce front udržuje počet (v poli *BackoutCount* deskriptoru zpráv) počtu případů, kdy k tomu dojde. Udržuje tento počet v deskriptoru každé zprávy, která je ovlivněna. Tento počet může poskytovat cenné informace o efektivitě aplikace. Zprávy s počtem odvolání, které se zvyšují v průběhu času, jsou opakovaně odmítány; navrhněte svou aplikaci tak, aby analyzovala příčiny těchto zpráv a ošetla tyto zprávy odpovídajícím způsobem.

z/OS Chcete-li, aby se počet odvolání přečkaly restartů správce front v produktu IBM MQ for z/OS, nastavte atribut **HardenGetBackout** na hodnotu MQQA_BACOUT_HARDENED; v opačném případě, pokud se má správce front restartovat, nezachová se pro každou zprávu přesný počet vrácení. Nastavení atributu tímto způsobem přidá penále za další zpracování.

V systémech IBM MQ for IBM i IBM i, Windows, UNIX and Linux systémy, počet vrácení vždy přežije restartování správce front.

z/OS Pokud také v produktu IBM MQ for z/OSodebírejte zprávy z fronty v rámci pracovní jednotky, můžete označit jednu zprávu tak, aby nebyla znovu zpřístupněna, je-li jednotka práce vrácena aplikací. Označená zpráva je zpracována, jako by byla načtena pod novou pracovní jednotkou. Můžete označit zprávu, která má vynechat odvolání pomocí volby MQGMO_MARK_SKIP_BACKUPL.(ve struktuře MQGMO), když použijete volání MQGET. Další informace o této technice naleznete v příručce “Vynechání odvolání” na stránce 766 .

Použití zpráv sestav k určování problémů

Vzdálený správce front nemůže hlásit chyby, jako je například selhání při vložení zprávy do fronty při provedení volání MQI, ale může vám odeslat zprávu s hlášením o tom, jak zprávu zpracoval.

Ve vaší aplikaci můžete vytvářet zprávy sestav (MQPUT) a také vybrat volbu jejich přijetí (v takovém případě jsou odesílány buď jinou aplikací, nebo správcem front).

Vytváření zpráv sestav

Zprávy hlášení umožňují aplikaci sdělit jiné aplikaci, že se nedokáže vypořádat se zprávou, která byla odeslána.

Pole *Report* však musí být nejprve analyzováno, aby bylo možné určit, zda aplikace, která zprávu odeslala, má zájem o informace o případných problémech. Poté, co jste určili, že je vyžadována zpráva sestavy, musíte se rozhodnout:

- Zda chcete zahrnout celou původní zprávu, pouze prvních 100 bajtů dat, nebo žádné z původní zprávy.
- Co se má provést s původní zprávou. Můžete ji vyřadit nebo nechat zahodit do fronty nedoručených zpráv.

- Zda je třeba také použít obsah polí *MsgId* a *CorrelId* .

Použijte pole *Feedback* , abyste označili důvod generování zprávy sestavy. Umístěte zprávy hlášení do fronty pro odpověď aplikace. Další informace najdete v tématu Zpětná vazba .

Vyžádání a příjem (MQGET) zpráv sestavy

Když odešlete zprávu do jiné aplikace, nebudete informováni o žádných problémech, dokud nedokončíte pole *Report* , abyste označili zpětnou vazbu, kterou požadujete. Dostupné volby naleznete v tématu Struktura pole sestavy .

Správci front vždy vloží zprávy sestav do fronty pro odpovědi aplikace a doporučuje se, aby vaše vlastní aplikace byly stejné. Použijete-li službu zpráv sestavy, zadejte název své odpovědi do fronty v deskriptoru zprávy vaší zprávy; jinak se volání MQPUT nezdaří.

Vaše aplikace musí obsahovat procedury, které monitorují vaši odpověď do fronty a zpracovávají zprávy, které do ní přicházejí. Nezapomeňte, že zpráva sestavy může obsahovat veškerou původní zprávu, prvních 100 bajtů původní zprávy nebo žádnou z původní zprávy.

Správce front nastaví pole *Feedback* ve zprávě sestavy tak, aby indikovalo příčinu chyby; například cílová fronta neexistuje. Vaše programy by měly provést totéž.

Další informace o zprávách sestav viz “Hlášení zpráv” na stránce 15.

Vzdáleně určené chyby

Při odesílání zpráv do vzdálené fronty, a to i v případě, že lokální správce front zpracoval volání MQI bez nalezení chyby, mohou jiné faktory ovlivnit způsob zpracování zprávy vzdáleným správcem front.

Například, fronta, kterou cílíte, může být plná, nebo nemusí existovat. Pokud má být vaše zpráva obsluhována jinými intermediačními správci front v přenosové cestě k cílové frontě, může některá z těchto informací nalézt chybu.

Problémy při doručování zprávy

Pokud se volání MQPUT nezdaří, můžete se pokusit o vložení zprávy do fronty znovu, vrátit ji odesilateli nebo ji umístit do fronty nedoručených zpráv.

Každá volba má své výhody, ale možná se nebudete chtít znovu pokusit o vložení zprávy z důvodu, že došlo k selhání příkazu MQPUT, protože cílová fronta byla plná. V této instanci umožňuje vložení fronty do fronty nedoručených zpráv později ji doručit do správné cílové fronty.

Zopakovat doručení zprávy

Před tím, než bude zpráva vložena do fronty nedoručených zpráv, se vzdálený správce front pokusí znovu vložit zprávu do fronty, pokud byly pro kanál nastaveny atributy *MsgRetryCount* a *MsgRetryInterval* , nebo pokud existuje uživatelský program opakování, který má být použit (jehož název je držen v poli atributu kanálu *MsgRetryExitId*).

Je-li pole *MsgRetryExitId* prázdné, použijí se hodnoty v atributech *MsgRetryCount* a *MsgRetryInterval* .

Není-li pole *MsgRetryExitId* prázdné, spustí se výstupní program tohoto názvu. Další informace o použití vlastních ukončovacích programů najdete v tématu “Kanály-uživatelské programy pro kanály systému zpráv” na stránce 926.

Vrátit zprávu odesílateli

Můžete vrátit zprávu odesílateli tak, že požádáte o vygenerování zprávy sestavy, aby zahrnovala všechny původní zprávy.

Podrobnosti o volbách zpráv sestav viz “Hlášení zpráv” na stránce 15 .

Použití fronty nedoručených zpráv (nedoručená zpráva)

Pokud správce front nemůže doručit zprávu, pokusí se vložit zprávu do fronty nedoručených zpráv. Tato fronta by měla být definována při instalaci správce front.

Vaše programy mohou používat frontu nedoručených zpráv stejným způsobem, jakým jej správce front používá. Název fronty nedoručených zpráv je možné najít otevřením objektu správce front (pomocí volání MQOPEN) a zjišťování informací o atributu **DeadLetterQName** (pomocí volání MQINQ).

Když správce front vloží do této fronty zprávu, přidá do zprávy záhlaví zprávy, jejíž formát je popsán strukturou záhlaví nedoručených zpráv (MQDLH); viz MQDLH-Dead-letter header. Do tohoto záhlaví patří název cílové fronty a důvod umístění zprávy do fronty nedoručených zpráv. Musí být odstraněn a problém musí být vyřešen před tím, než je zpráva vložena do zamýšlené fronty. Dále správce front změní pole *Format* deskriptoru zpráv (MQMD) tak, aby indikovalo, že zpráva obsahuje strukturu MQDLH.

Struktura MQDLH

Doporučuje se přidat strukturu MQDLH ke všem zprávám, které jste umístili do fronty nedoručených zpráv. Pokud však chcete použít obslužnou rutinu nedoručených zpráv, která je poskytována některými produkty IBM MQ , je třeba do zpráv přidat strukturu MQDLH.

Přidání záhlaví do zprávy může pro frontu nedoručených zpráv příliš dlouhé, takže se vždy ujistěte, že jsou zprávy kratší než maximální velikost povolená pro frontu nedoručených zpráv, a to alespoň hodnotou konstanty MQ_MSG_HEADER_LENGTH. Maximální velikost zpráv povolených ve frontě je určena hodnotou atributu **MaxMsgLength** ve frontě. U fronty nedoručených zpráv se ujistěte, že tento atribut je nastaven na maximum povolené správcem front. Pokud vaše aplikace nemůže doručit zprávu a zpráva je příliš dlouhá na to, aby byla vložena do fronty nedoručených zpráv, postupujte podle pokynů uvedených v popisu struktury MQDLH.

Ujistěte se, že je fronta nedoručených zpráv monitorována a že všechny zprávy přicházející do tohoto stavu se zpracují. Obslužná rutina fronty nedoručených zpráv je spouštěna jako dávkový obslužný program a lze ji použít k provádění různých akcí ve vybraných zprávách ve frontě nedoručených zpráv. Další podrobnosti viz “Zpracování fronty nedoručených zpráv” na stránce 1019.

Je-li konverze dat nezbytná, převede správce front informace v záhlaví, když použijete volbu MQGMO_CONVERT na volání MQGET. Je-li proces, který zprávu vkládá, je MCA, je za záhlavím následován veškerý text původní zprávy.

Zprávy vkládané do fronty nedoručených zpráv mohou být zkráceny v případě, že jsou pro tuto frontu příliš dlouhé. Možným označením této situace jsou zprávy ve frontě nedoručených zpráv, které mají stejnou délku jako hodnota atributu **MaxMsgLength** ve frontě.

Zpracování fronty nedoručených zpráv

Tyto informace obsahují informace o programovacím rozhraní generální-použití při použití zpracování fronty nedoručených zpráv.

Zpracování fronty nedoručených zpráv závisí na požadavcích lokálního systému, ale při sestavování specifikace zvažte následující skutečnosti:

- Zpráva může být identifikována jako záhlaví fronty nedoručených zpráv, protože hodnota pole formátu v deskriptoru MQMD je MQFMT_DEAD_LETTER_HEADER.
- Pokud má produkt MCA v produktu IBM MQ for z/OS za použití CICStuto zprávu do fronty nedoručených zpráv, pole *PutApplType* je MQAT_CICSa pole *PutApplName* je *ApplId* v systému CICS , za nímž následuje název transakce MCA.
- Příčina pro zprávu, která má být směrována do fronty nedoručených zpráv, je obsažena v poli *Reason* záhlaví fronty nedoručených zpráv.
- Hlavička fronty nedoručených zpráv obsahuje podrobnosti o názvu cílové fronty a názvu správce front.
- Hlavička fronty nedoručených zpráv obsahuje pole, která musí být obnovena v deskriptoru zpráv před tím, než je zpráva vložena do cílové fronty. Patří mezi ně:

  1. *Encoding*

2. *CodedCharSetId*
3. *Format*

- Deskriptor zprávy je stejný jako PUT původní aplikací, s výjimkou tří zobrazených polí (Encoding, CodedCharSetIda Format).

Vaše aplikace fronty nedoručených zpráv musí provést jednu nebo více z následujících možností:

- Prověřte pole *Reason* . Je možné, že zpráva byla vložena MCA z následujících důvodů:
  - Zpráva byla delší než maximální velikost zprávy pro kanál.

    Příčina: MQRC_MSG_TOO_BIG_FOR_CHANNEL
  - Zprávu nelze zařadit do cílové fronty.

    Důvodem je libovolný kód příčiny MQRC_ *, který může být vrácen operací MQPUT .
  - Uživatelská procedura požadovala tuto akci

    Kód příčiny je dodán uživatelskou procedurou nebo výchozí hodnota MQRC_SUPPRESSED_BY_EXIT.
- Pokuste se předat zprávu do zamýšleného místa určení, pokud je to možné.
- Zachovejte zprávu po určitou dobu před vyřazením, je-li určena příčina zneužití, ale ne okamžitě opravitelná tabulka.
- Pokyny pro administrátory opravujte o problémech tam, kde je to určeno.
- Vyřaďte zprávy, které jsou poškozené nebo které nejsou zpracovatelné.

Existují dva způsoby, jak se vypořádat se zprávami, které jste se zotavili z fronty zablokovaných dopisů:

1. Je-li zpráva pro lokální frontu:
   - Provést všechny překlady kódu nezbytné pro extrakci dat aplikace
   - Provést převody kódu na těchto datech, pokud se jedná o lokální funkci
   - Vložení výsledné zprávy do lokální fronty se všemi podrobnostmi obnoveného deskriptoru zpráv
2. Je-li zpráva určena pro vzdálenou frontu, vložte ji do fronty.

Informace o tom, jak jsou obsluhovány nedoručené zprávy v distribuovaném prostředí s frontou, najdete v tématu Co se děje, když nelze zprávu doručit?.

Programování multicast

Tyto informace použijte k seznámení se s úlohami programování výběrového vysílání produktu IBM MQ , jako je připojení ke správci front a hlášení výjimek.

Výběrové vysílání produktu IBM MQ bylo navrženo tak, aby bylo pro uživatele co nejtransparentnější, a přesto je kompatibilní s existujícími aplikacemi. Definování objektu COMMINFO a nastavení parametrů **MCAST** a **COMMINFO** objektu TOPIC znamená, že existující aplikace produktu IBM MQ nevyžadují výrazné přepsání pro použití výběrového vysílání. Mohou však existovat určitá omezení (více informací viz “Výběrové vysílání a MQI” na stránce 1020) a některé bezpečnostní otázky (viz Zabezpečení výběrového vysílání , kde získáte další informace).

Výběrové vysílání a MQI

Prostřednictvím těchto informací lze porozumět hlavním konceptům rozhraní MQI (Message Queue Interface) a způsobu, jakým souvisejí s výběrovým vysíláním produktu IBM MQ .

Odběry výběrového vysílání jsou netrvanlivé; protože nejsou k dispozici žádné fyzické fronty, neexistuje místo pro uložení zpráv offline, které byly vytvořeny trvalnými odběry.

Poté, co aplikace bude přihlášena k odběru tématu výběrového vysílání, je mu přidělen popisovač objektu, který může spotřebovat, nebo MQGET, jako by se jednalo o popisovač fronty. To znamená, že jsou podporovány pouze spravované odběry výběrového vysílání (odběry vytvořené pomocí MQSO_MANAGED), což znamená, že není možné vytvořit odběr a "bod" zprávy ve frontě. To znamená, že zprávy musí být spotřebovávány z manipulátoru objektu vráceného při volání odběru. Na straně klienta jsou zprávy

uloženy do vyrovnávací paměti zpráv, dokud je klient nespotřebuje. Další informace naleznete v sekci MessageBuffer konfiguračního souboru klienta . Pokud klient nepodrží krok s rychlostí publikování, budou zprávy vyřazeny podle potřeby, přičemž nejstarší zprávy byly vyřazeny jako první.

Obvykle se jedná o rozhodnutí administrace, zda aplikace používá výběrové vysílání nebo ne, určené nastavením atributu MCAST objektu TOPIC. Pokud musí publikační aplikace zajistit, aby výběrové vysílání nebylo použito, může použít volbu `MQOO_NO_MULTICAST` . Podobně i odběratelská aplikace může zajistit, aby výběrové vysílání nebylo používáno při přihlašování k odběru s použitím volby `MQSO_NO_MULTICAST` .

Výběrové vysílání produktu IBM MQ podporuje použití selektorů zpráv. Selektor je používán aplikací k registraci svého zájmu pouze v těch zprávách s vlastnostmi, které splňují dotaz SQL92 , který představuje řetězec výběru. Další informace o selektorech zpráv viz “Selektory.” na stránce 25.

V následující tabulce jsou uvedeny všechny hlavní koncepce MQI a informace o tom, jak souvisí s výběrovým vysíláním:

Tabulka 143. Koncepce MQI a jejich souvislost s výběrovým vysíláním

Koncepce MQI	**Akce při pokusu pomocí výběrového vysílání**	**Kód příčiny**
Vložení zprávy o nulové délce	Odmítnuto	2005 (07D5) (RC2005): CHYBA MQRC_BUFFER_LENGTH_ERROR
Seskupení	Odmítnuto	2046 (07FE) (RC2046): CHYBA MQRC_OPTIONS_ERROR
Segmentace	Odmítnuto	2443 (098B) (RC2443): MQRC_SEGMENTATION_NOT_ALLOWED
Distribuční seznamy	Odmítnuto	2154 (086A) (RC2154): MQRC_REC_PRESENT_ERROR
MQINQ	Odmítnuto pro zpracování témat: MQINQ a MQSET témat nejsou podporovány.	2038 (07F6) (RC2038): MQRC_NOT_OPEN_FOR_INQUIRE
MQINQ	Přijato pro spravovaný popisovač. Pouze `Aktuální hloubka` může být dotazovaná.	• Je-li hodnota Aktuální hloubka, pak neexistuje žádný vhodný kód příčiny. • Je-li hodnota jiná než Aktuální hloubka, kód příčiny je 2067 (0813) (RC2067): MQRC_SELECTOR_ERROR.
MQSET	Zamítnuto pro všechny popisovače.	2040 (07F8) (RC2040): MQRC_NOT_OPEN_FOR_SET
Transakce (XA nebo ne)	Odmítnuto	2072 (0818) (RC2072): MQRC_SYNCPOINT_NOT_AVAILABLE
Procházení zpráv	Odmítnuto	2036 (07F4) (RC2036): MQRC_NOT_OPEN_FOR_BROWSE
Uzamknout zprávy	Odmítnuto	2046 (07FE) (RC2046): CHYBA MQRC_OPTIONS_ERROR
Procházet značkou	Odmítnuto	2036 (07F4) (RC2036): MQRC_NOT_OPEN_FOR_BROWSE
Předat kontext	Odmítnuto	2046 (07FE) (RC2046): CHYBA MQRC_OPTIONS_ERROR

Tabulka 143. Koncepce MQI a jejich souvislost s výběrovým vysíláním (pokračování)		
Koncepce MQI	**Akce při pokusu pomocí výběrového vysílání**	**Kód příčiny**
MQPUT1	Zamítnuto. Je neplatné pokusit se MQPUT1 použít pouze pro téma Výběrové vysílání.	2560 (0A00) (RC2560): MQRC_MULTICAST_ONLY
trvalý odběr	Zamítnuto, je-li téma označeno jako "Pouze výběrové vysílání", jinak se provede odběr jiného typu než Výběrové vysílání.	2436 (0984) (RC2436): MQRC_DURABILITY_NOT_ALLOWED
TopicString > 255	Zamítnuto. Je-li řetězec tématu delší než 255 znaků, je odmítnut v klientovi.	2425 (0979) (RC2425): MQRC_TOPIC_STRING_ERROR
Provedené nespravované odběry	Zamítnuto, je-li téma označeno jako "Pouze výběrové vysílání", jinak se provede odběr jiného typu než Výběrové vysílání.	2046 (07FE) (RC2046): CHYBA MQRC_OPTIONS_ERROR
MQPMO_NOT_OWN_SUBS	Odmítnuto	2046 (07FE) (RC2046): CHYBA MQRC_OPTIONS_ERROR

Následující položky se rozbalí v některých koncepcích MQI z předchozí tabulky a poskytují informace o některých konceptech MQI, které nejsou v tabulce:

Trvalost zpráv
V případě odběratelů výběrového vysílání jsou trvalé zprávy od vydavatele doručovány neobnovitelným způsobem.

Zkrácení zprávy
Oříznutí zprávy je podporováno, což znamená, že je možné, aby aplikace mohla:

1. Zadejte příkaz MQGET.
2. Získání objektu MQRC_TRUNCATED_MSG_FAILED.
3. Přidělte větší vyrovnávací paměť.
4. Znovu zadejte příkaz MQGET a načtěte zprávu.

Vypršení platnosti odběru
Vypršení platnosti odběru není podporováno. Jakýkoli pokus o nastavení vypršení platnosti je ignorován.

Vysoká dostupnost pro výběrové vysílání

Tyto informace použijte k pochopení nepřetržitého provozu výběrového vysílání produktu IBM MQ typu peer; i když se produkt IBM MQ připojuje ke správci front produktu IBM MQ , neprocházejí zprávy tímto správcem front.

Přestože musí být vytvořeno připojení ke správci front, aby bylo možné objekt tématu výběrového vysílání MQOPEN nebo MQSUB MQUB, neprotékat prostřednictvím správce front zprávy samotné. Proto je po dokončení operace MQOPEN nebo MQSUB na objektu tématu výběrového vysílání možné pokračovat ve vysílání zpráv výběrového vysílání, a to i v případě, že připojení ke správci front bylo ztraceno. Existují dva režimy provozu:

Je vytvořeno běžné připojení ke správci front
Komunikaci multicast je možné, když existuje připojení ke správci front. Pokud připojení selže, jsou použita běžná pravidla MQI, například; příkaz MQPUT pro obsluhu objektů výběrového vysílání vrátí hodnotu 2009 (07D9) (RC2009): MQRC_CONNECTION_BROKEN.

Připojuje se znovu připojení klienta ke správci front.
Komunikace multicast je možná i během reconnection cyklu. To znamená, že i když došlo k přerušení připojení ke správci front, tím není ovlivněno vložení zpráv výběrového vysílání a příjem zpráv výběrového vysílání. Klient se pokusí znovu připojit ke správci front, a pokud toto opětovné připojení selže, stane se popisovač připojení přerušený a všechna volání MQI, včetně těch s výběrovým vysíláním, selžou. Další informace naleznete v tématu: Automatické opětovné připojení klienta

Pokud některá aplikace explicitně vydá příkaz MQDISC, budou všechny odběry výběrového vysílání a manipulátory objektů zavřeny.

Kontinuální nepřetržitá operace typu

Jednou z výhod komunikace typu P2P mezi klienty je to, že zprávy nemusí procházet správcem front. Pokud se tedy připojení ke správci front přeruší, bude přenos zpráv pokračovat. Pro požadavky na souvislé zprávy v tomto režimu se vztahují následující omezení:

- Připojení musí být vytvořeno pomocí jedné z voleb MQCNO_RECONICONNECT_ * pro průběžnou operaci. Tento proces znamená, že ačkoli komunikační relace může být porušena, není aktuální manipulátor připojení přerušený a nachází se v novém připojeném stavu. Dojde-li k selhání nového připojení, je nyní popisovač připojení přerušen, což brání všem dalším voláním MQI.
- V tomto režimu jsou podporovány pouze příkazy MQPUT, MQGET, MQINQ a Async Consume. Příkazové příkazy MQOPEN, MQCLOSE nebo MQDISC vyžadují opětovné připojení ke správci front, aby bylo dokončeno.
- Stav toků ke správci front se zastaví; jakýkoli stav ve správci front proto může být zastaralý nebo chybějící. To znamená, že klienti mohou odesílat a přijímat zprávy a že ve správci front není znám žádný stav. Další informace naleznete v tématu: Monitorování aplikace výběrového vysílání

Převod dat v rozhraní MQI pro systém zpráv výběrového vysílání

Pomocí těchto informací můžete porozumět způsobu, jakým převod dat pracuje pro systém zpráv výběrového vysílání IBM MQ .

Výběrové vysílání produktu IBM MQ je sdíleno, bezspojový protokol, a proto není možné, aby každý klient mohl provádět specifické požadavky na převod dat. Každý klient přihlášený k odběru stejného vícesměrového vysílání přijímá stejná binární data; proto, je-li zapotřebí konverze dat IBM MQ , provede se konverze lokálně na každém klientovi.

Data jsou převedena na klienta pro provoz výběrového vysílání produktu IBM MQ . Je-li zadána volba **MQGMO_CONVERT** , převod dat se provádí podle požadavku. Uživatelsky definované formáty potřebují uživatelskou proceduru pro převod dat nainstalovanou na klientovi; informace o tom, které knihovny jsou nyní v balících klienta a serveru, najdete v příručce “Zápis uživatelských procedur pro převod dat” na stránce 948 .

Informace o administraci převodu dat naleznete v tématu Povolení převodu dat pro systém zpráv výběrového vysílání.

Další informace o konverzi dat najdete v tématu Převod dat.

Další informace o uživatelských procedurách pro převod dat a produktu `ClientExitPath`naleznete v části ClientExitPath stanza konfiguračního souboru klienta.

Vykazování výjimek výběrového vysílání

Tyto informace použijte, chcete-li se dozvědět více o obslužných programech výběrového vysílání IBM MQ a hlášení výjimek výběrového vysílání produktu IBM MQ .

Výběrové vysílání produktu IBM MQ pomáhá s určováním problémů voláním obslužné rutiny událostí za účelem hlášení událostí výběrového vysílání, které jsou ohlášeny za použití standardního mechanizmu obslužné rutiny událostí produktu IBM MQ .

Jednotlivé události výběrového vysílání mohou vést k volání více než jedné události IBM MQ , protože může existovat více obslužných rutin `MQHCONN` připojení pomocí stejného vysílače výběrového vysílání nebo přijímače. Každá výjimka výběrového vysílání však způsobí, že bude volána pouze jedna obslužná rutina událostí pro připojení produktu IBM MQ .

Konstanta IBM MQ `MQCBDO_EVENT_CALL` umožňuje aplikacím registrovat zpětné volání tak, aby přijímaly pouze události produktu IBM MQ , a aplikace `MQCBDO_MC_EVENT_CALL` umožňuje aplikacím zaregistrovat zpětné volání, aby bylo možné přijímat pouze události výběrového vysílání. Jsou-li použity obě konstanty, jsou přijaty oba typy událostí.

Vyžádání událostí výběrového

Události výběrového vysílání produktu IBM MQ používají konstantu `MQCBDO_MC_EVENT_CALL` v poli `cbd.Options` . Následující příklad ukazuje, jak požadovat události výběrového vysílání:

```
cbd.CallbackType     = MQCBT_EVENT_HANDLER;
cbd.Options          = MQCBDO_MC_EVENT_CALL;
cbd.CallbackFunction = EventHandler;
MQCB(Hcon,MQOP_REGISTER,&cbd,MQHO_UNUSABLE_HOBJ,NULL,NULL,&CompCode,&Reason);
```

Je-li zadána volba `MQCBDO_MC_EVENT_CALL` pro pole `cbd.Options` , obslužná rutina událostí odešle pouze události výběrového vysílání IBM MQ místo událostí na úrovni připojení. Pro požadavek, aby byly do obslužné rutiny událostí odeslány oba typy událostí, musí aplikace zadat konstantu `MQCBDO_EVENT_CALL` do pole `cbd.Options` stejně jako konstantu `MQCBDO_MC_EVENT_CALL` , jak ukazuje následující příklad:

```
cbd.CallbackType     = MQCBT_EVENT_HANDLER;
cbd.Options          = MQCBDO_EVENT_CALL | MQCBDO_MC_EVENT_CALL
cbd.CallbackFunction = EventHandler;
MQCB(Hcon,MQOP_REGISTER,&cbd,MQHO_UNUSABLE_HOBJ,NULL,NULL,&CompCode,&Reason);
```

Není-li použit ani jeden z těchto konstant, budou do obslužné rutiny událostí odeslány pouze události úrovně připojení.

Další informace o hodnotách pro pole `Options` naleznete v části Volby (MQLONG).

Formát událostí výběrového vysílání

IBM MQ Výjimky výběrového vysílání obsahují některé podpůrné informace, které jsou vráceny v argumentu **`Buffer`** funkce zpětného volání. Ukazatel **`Buffer`** ukazuje na pole ukazatelů a v poli `MQCBC.DataLength` je uvedena velikost pole v bajtech. První prvek pole vždy ukazuje na krátký textový popis události. Další parametry mohou být dodány v závislosti na typu události. V následující tabulce jsou uvedeny výjimky:

Tabulka 144. Popisy kódů událostí výběrového vysílání

Kód události	Popis	Další data
SRÁŽKA MQMTEV_PACKET_LOSS	Neopravitelná ztráta paketů	Počet ztracených paketů
ČASOVÝ LIMIT MQMCAV_HEARTBEAT_TIMEOUT	Dlouhá absence řídicího paketu prezenčního signálu	Není k dispozici
KONFLIKT MQMTEV_VERSION_CONFLICT	Přijímání novějších paketů verze protokolu	Není k dispozici
MQMCEVA_RELIABILITY	Různé režimy spolehlivosti vysílače a přijímače	Není k dispozici
UZAVŘÍT MQMCEV_CLOSED_TRANS	Přenos tématu je uzavřen jedním zdrojem	Není k dispozici
CHYBA MQMTEV_STREAM_ERROR	Zjištěna chyba v proudu	Není k dispozici
ZDROJ MQMCEV_NEW_SOURCE	Nový zdroj začne vysílat na téma	Zdrojová struktura
MAČ_PŘÍJEMCE_PŘIJMU_MQUT_DATA_OŘÍZNUTO	Odebrané pakety z hodnoty PacketQ kvůli vypršení času nebo vypršení platnosti prostoru.	Počet ořízených paketů
SKONČIT PLATNOST: MQMCEV_PACKET_LOSS_NICK_EXPIRE	Neopravitelná ztráta paketu kvůli vypršení platnosti NACK	Počet ztracených paketů
BYL PŘEKROČEN PARAMETR MQMVAR_ACK_RETRIES_	Pakety odebrané z historie po překročení **max_ack_retries**	Počet odebraných paketů
MQMCEV_STREAM_SUSPEND_NACK	NACK byly pozastaveny na proud akceptovaný tímto tématem	Pozastavit ID proudu Doba, po kterou je proud pozastaven
MQMCEV_STREAM_RESUME_NACK	NACKs byly obnoveny poté, co byly pozastaveny na proudu	ID proudu
VYLOUČENÝ PROUD MQMTEV_STREAM_EX	Proud přijatý tímto tématem byl zamítnut v důsledku požadavku na vyloučení	ID proudu
ZPRÁVA MQMTEV_FIRST_MESSAGE	První zpráva ze zdroje	Číslo zprávy.
SELHÁNÍ FUNKCE MQMCEV_LACE_JOIN_FAILURE	Spuštění relace zpožděného spojení se nezdařilo	Není k dispozici
SKRYTÍ MQMEV_MESSAGE_LOSS	Neopravitelná ztráta zpráv	Počet ztracených zpráv
SELHÁNÍ MQMTEV_SEND_PACKET_FAILURE	Vysílač výběrového vysílání selhal při odesílání paketu výběrového vysílání	Není k dispozici
ZPOŽDĚNÍ MQMEV_REPACE_	Přijímač výběrového vysílání neobdržel paket opravy pro neprovedený NAK	Není k dispozici
MQMCEV_MEMORY_ALERT_ON	Vyrovnávací paměti příjmu zásobníku se zaplňují	Procento využití fondu vyrovnávacích pamětí
MQMCEV_MEMORY_ALERT_OFF	Vyrovnávací paměti příjmu zásobníku jsou mimo provoz.	Procento využití fondu vyrovnávacích pamětí

Tabulka 144. Popisy kódů událostí výběrového vysílání (pokračování)

Kód události	Popis	Další data
MQMCE_V_NACK_ALERT_ON	Rychlost požadavku na opravu paketu byla dosažena horní mez	Aktuální četnost požadavků na opravu v paketech za sekundu
MQMCEV_NACK_ALERT_OFF	Rychlost požadavků paketů na opravu zásobníku je mimo provoz na normální úroveň	Aktuální četnost požadavků na opravu v paketech za sekundu
MQMETEV_REPACE_ALERT_ON	Rychlost odesílání paketů pro opravu paketu pro opravu vysílače dosáhla horní meze	Není k dispozici
MQMCEV_REPAIR_ALERT_OFF	Rychlost odeslání paketu pro opravu vysílače je mimo provoz.	Není k dispozici
MQMTEV_SHM_DEST_UNUSABLE	Oblast sdílené paměti použitá místem určení tématu vysílače byla zjištěna jako nepoužitelná.	Není k dispozici
MQMTEV_SHM_PORT_UNUSABLE	Port sdílené paměti použitý instancí příjemce byl zjištěn jako nepoužitelný	Není k dispozici
SELHÁNÍ MQMCEV_CCT_GETTIME_FAILED	Čas získání z koordinovaného času klastru se nezdařil	Není k dispozici
SELHÁNÍ PŘI SELHÁNÍ MQMTEV_DEST_INTERFACE_FAILURE	Síťové rozhraní použité místem určení tématu vysílače se nezdařilo a záložní síťové rozhraní je nedostupné.	
MQMCEV_DEST_INTERFACE_FAILOVER	Síťové rozhraní použité místem určení tématu vysílače se nezdařilo a úspěšné překonání selhání na jiné rozhraní bylo dokončeno.	
MQMTEV_PORT_INTERFACE-SELHÁNÍ	Síťové rozhraní použité příjemcem rmmPort se nezdařilo a záložní síťové rozhraní je nedostupné (nebo se také nezdařilo)	KonfiguraceRMM
MQMCEV_PORT_INTERFACE_FAILOVER	Síťové rozhraní použité příjemcem rmmPort se nezdařilo a úspěšné překonání selhání na jiné rozhraní bylo dokončeno	KonfiguraceRMM

Cování v C

Všimněte si informací v následujících sekcích při kódování programů IBM MQ v C.

- “Parametry volání MQI” na stránce 1027
- “Parametry s nedefinovaným datovým typem” na stránce 1027
- “Datové typy” na stránce 1027
- “Manipulace s binárními řetězci” na stránce 1027
- “Manipulace se znakovými řetězci” na stránce 1028

- “Počáteční hodnoty pro struktury” na stránce 1028
- “Počáteční hodnoty pro dynamické struktury” na stránce 1028
- “Použít z C++” na stránce 1029

Parametry volání MQI

Parametry, které jsou *pouze pro vstup* a typu MQHCONN, MQBOBJ, MQHMSG nebo MQLONG, jsou předávány hodnotou; pro všechny ostatní parametry je hodnota parametru předávána hodnotou parametru *address* .

Ne všechny parametry, které jsou předávány podle adresy, je třeba zadat pokaždé, když je vyvolána funkce. Není-li určitý parametr požadován, lze jako parametr při vyvolání funkce zadat ukazatel Null jako parametr na místo adresy dat parametru. Parametry, pro které je to možné, jsou identifikovány v popisech volání.

Jako hodnotu funkce se nevrací žádný parametr; v terminologii C to znamená, že všechny funkce vrací neobsazenou hodnotu.

Atributy funkce jsou definovány proměnnou makra MQENTRY; hodnota této proměnné makra závisí na daném prostředí.

Parametry s nedefinovaným datovým typem

Každý z funkcí MQGET, MQPUT a MQPUT1 má parametr **Buffer** , který má nedefinovaný datový typ. Tento parametr se používá k odesílání a přijímání dat zprávy aplikace.

Parametry tohoto řazení jsou zobrazeny v příkladech C jako pole MQBYTE. Parametry můžete deklarovat tímto způsobem, ale je obvykle výhodnější deklarovat je jako strukturu, která popisuje rozvržení dat ve zprávě. Parametr funkce je deklarovaný jako ukazatel na neobsazený, a proto lze jako parametr ve vyvolání funkce určit adresu jakýchkoli dat.

Datové typy

Všechny datové typy jsou definovány pomocí příkazu `typedef` .

Pro každý datový typ je také definován odpovídající datový typ ukazatele. Název datového typu ukazatele je název elementárních nebo strukturních datových typů s předponou písmenem P, která označuje ukazatel. Atributy ukazatele jsou definovány proměnnou makra MQPOINTER; hodnota této proměnné makra závisí na daném prostředí. Následující kód ukazuje, jak deklarovat datové typy ukazatele:

```
#define MQPOINTER                  /* depends on environment */
...
typedef MQLONG  MQPOINTER PMQLONG;  /* pointer to MQLONG */
typedef MQMD   MQPOINTER PMQMD;     /* pointer to MQMD */
```

Manipulace s binárními řetězci

Řetězce binárních dat jsou deklarovány jako jeden z datových typů MQBYTEn.

Kdykoli kopírujete, porovnávají nebo nastavují pole tohoto typu, použijte funkce C `memcpy`, `memcmp`nebo `memset`:

```
#include <string.h>
#include "cmqc.h"

MQMD MyMsgDesc;

memcpy(MyMsgDesc.MsgId,           /* set "MsgId" field to nulls    */
       MQMI_NONE,                 /* ...using named constant      */
       sizeof(MyMsgDesc.MsgId));

memset(MyMsgDesc.CorrelId,        /* set "CorrelId" field to nulls */
```

```
       0x00,                       /* ...using a different method   */
       sizeof(MQBYTE24));
```

Nepoužívejte řetězcové funkce strcpy, strcmp, strncpy nebo strncmp, protože tyto funkce nefungují správně s daty deklarovanými jako MQBYTE24.

Manipulace se znakovými řetězci

Když správce front vrátí do aplikace znaková data, správce front vždy vycpávku znaková data s mezerami do definované délky pole. Správce front nevrací řetězce ukončené hodnotou null, ale můžete je použít ve svém vstupu. Proto při kopírování, porovnání nebo zřetězení takových řetězců použijte řetězec funkcí strncpy, strncmp nebo strncat.

Nepoužívejte funkce řetězce, které vyžadují, aby řetězec byl ukončen znakem hex 00 (strcpy, strcmp a strcat). Also, do not use the function strlen to determine the length of the string; use instead the sizeof function to determine the length of the field.

Počáteční hodnoty pro struktury

Soubor začlenění <cmqc.h> definuje různé proměnné maker, které můžete použít k poskytnutí výchozích hodnot pro struktury při deklarování instancí těchto struktur. Tyto proměnné maker mají názvy ve tvaru MQxxx_DEFAULT, kde MQxxx představuje název struktury. Použijte je takto:

```
MQMD   MyMsgDesc = {MQMD_DEFAULT};
MQPMO  MyPutOpts = {MQPMO_DEFAULT};
```

Pro některá znaková pole rozhraní MQI definuje konkrétní platné hodnoty (například pro pole *StrucId* nebo pole *Format* v produktu MQMD). Pro každou z platných hodnot jsou k dispozici dvě proměnné makra:

- Jedna makroproměnná definuje hodnotu jako řetězec s délkou, s výjimkou implikované hodnoty null, která přesně odpovídá definované délce pole. Například symbol ¬ představuje prázdný znak:

  ```
  #define MQMD_STRUC_ID "MD¬¬"
  #define MQFMT_STRING "MQSTR¬¬¬"
  ```

  Použijte tento formulář s funkcemi memcpy a memcmp.
- Další proměnná makra definuje hodnotu jako pole typu char; název této proměnné makra je název řetězce tvořená řetězcem s příponou _ARRAY. Příklad:

  ```
  #define MQMD_STRUC_ID_ARRAY 'M','D','¬','¬'
  #define MQFMT_STRING_ARRAY 'M','Q','S','T','R','¬','¬','¬'
  ```

  Tento formulář použijte k inicializaci pole, když je instance struktury deklarována s hodnotami, které jsou odlišné od proměnné poskytnuté proměnnou makra MQMD_DEFAULT.

Počáteční hodnoty pro dynamické struktury

Když se požaduje proměnný počet instancí struktury, jsou instance obvykle vytvářeny v hlavní paměti získávané dynamicky pomocí funkcí calloc nebo malloc.

Chcete-li inicializovat pole v těchto strukturách, doporučuje se následující technika:

1. Deklarujte instanci struktury pomocí příslušné makro proměnné MQxxx_DEFAULT k inicializaci struktury. Tato instance se stane *modelem* pro jiné instance:

   ```
   MQMD ModelMsgDesc = {MQMD_DEFAULT};
                                     /* declare model instance */
   ```

Kódováním statických nebo automatických klíčových slov v deklaraci poskytnete instanci modelu statickou nebo dynamickou dobu životnosti podle potřeby.

2. K získání paměti pro dynamickou instanci struktury použijte funkce calloc nebo malloc:

```
PMQMD InstancePtr;
InstancePtr = malloc(sizeof(MQMD));
                                    /* get storage for dynamic instance */
```

3. Použijte funkci memcpy ke zkopírování instance modelu do dynamické instance:

```
memcpy(InstancePtr,&ModelMsgDesc,sizeof(MQMD));
                                    /* initialize dynamic instance */
```

Použít z C++

V případě programovacího jazyka C + + obsahují hlavičkové soubory následující další příkazy, které jsou zahrnuty pouze v případě použití kompilátoru C + +:

```
#ifdef __cplusplus
  extern "C" {
#endif

/* rest of header file */

#ifdef __cplusplus
  }
#endif
```

Windows Kódování v produktu Visual Basic

Informace, které je třeba zvážit při kódování programů IBM MQ v produktu Microsoft Visual Basic. Produkt Visual Basic je podporován pouze v systému Windows.

Poznámka: V prostředí IBM WebSphere MQ 7.0se mimo prostředí .NET podpora pro Visual Basic (VB) stabilizovala na úrovni IBM WebSphere MQ 6.0 . Většina nových funkcí přidaných do produktu IBM WebSphere MQ 7.0 nebo pozdějších verzí není k dispozici pro aplikace VB. Pokud programuji v VB.NET, použijte třídy IBM MQ pro .NET. Další informace viz Vývoj aplikací .NET.

V 9.0.0 V produktu IBM MQ 9.0je podpora produktu Microsoft Visual Basic 6.0 zamítnuta. Třídy IBM MQ pro .NET jsou doporučenými náhradními technologiemi.

Chcete-li se vyhnout nechtěnému překladu binárních dat, které procházejí mezi Visual Basic a IBM MQ, použijte definici MQBYTE místo MQSTRING. CMQB.BAS definuje několik nových typů MQBYTE, které jsou ekvivalentní definice typu C byte a používají je v rámci struktur produktu IBM MQ . Například pro strukturu MQMD (deskriptor zprávy) je položka MsgId (identifikátor zprávy) definována jako MQBYTE24.

Visual Basic nemá datový typ ukazatele, takže odkazy na jiné datové struktury IBM MQ jsou raději offsetem než ukazatelem. Deklarujte složenou strukturu skládající se ze dvou struktur komponent a určete složenou strukturu na volání. IBM MQ support for Visual Basic provides an MQCONNXAny call to make this possible and allow client applications to specify the channel properties on a client connection. Přijímá netypované struktury (MQCNOCD) místo typické struktury MQCNO.

Struktura MQCNOCD je složená struktura skládající se z objektu MQCNO a za ním příkaz MQCD. Tato struktura je deklarována v souboru záhlaví uživatelských procedur CMQXB. Pomocí rutiny MQCNOCD_DEFAULTS inicializujte strukturu MQCNOCD. K dispozici je ukázka volání MQCONNX (amqscnxb.vbp).

MQCONNXAny má stejné parametry jako MQCONNX, s výjimkou toho, že parametr **ConnectOpts** je deklarován jako typ Any, spíše než datový typ MQCNO. To umožňuje, aby funkce přijala buď strukturu MQCNO, nebo strukturu MQCNOCD. Tato funkce je deklarována v hlavním souboru záhlaví CMQB.

Související pojmy

“Příprava programů produktu Visual Basic v produktu Windows” na stránce 998
Informace, které je třeba zvážit při používání programů produktu Microsoft Visual Basic v systému Windows.

Související odkazy

“Propojení aplikací produktu Visual Basic s kódem produktu IBM MQ MQI client” na stránce 882
You can link Microsoft Visual Basic applications with the IBM MQ MQI client code on Windows.

Kódování v jazyce COBOL

Všimněte si informací v následující sekci při kódování programů IBM MQ v jazyce COBOL.

Pojmenované konstanty

Názvy konstant jsou zobrazeny obsahující znak podtržítka (_) jako součást názvu. V COBOLu, musíte použít znak pomlčky (-) místo podtržítka. Konstanty, které mají hodnoty znakových řetězců, používají jako oddělovač řetězců znak apostrofu ('). Má-li kompilátor přijmout tento znak, použijte volbu kompilátoru APOST.

Kopírovaná soubor CMQV obsahuje deklarace pojmenovaných konstant s položkami level-10 . Chcete-li použít konstanty, deklarujte explicitně položku level-01 , pak použijte příkaz COPY ke kopírování v deklaracích konstant:

```
WORKING-STORAGE SECTION.
01 MQM-CONSTANTS.
COPY CMQV.
```

Tato metoda však způsobí, že se konstanty zabírají v programu i v případě, že na ně není odkazováno. Pokud jsou konstanty zahrnuty do mnoha samostatných programů v rámci stejné jednotky spuštění, bude existovat více kopií konstant; to může mít za následek použití významného množství hlavní paměti. Tomuto problému se můžete vyhnout přidáním klauzule GLOBAL do deklarace level-01 :

```
* Declare a global structure to hold the constants
01 MQM-CONSTANTS GLOBAL.
COPY CMQV.
```

Toto alokuje paměť pouze pro *jednu* sadu konstant v rámci jednotky spuštění; konstanty se však mohou odvolávkou na *libovolný* program v rámci spuštěné jednotky, nejen program, který obsahuje deklaraci level-01 .

Zajištění zarovnání struktury

Měli byste dbát na to, aby struktury IBM MQ , které jsou předávány ke spuštění na volání MQ , byly zarovnány na hranicích slova. Hranice slova je 4 bajty pro 32bitové procesy, 8 bajtů pro 64 bitové procesy a 16 bajtů pro 128bitové procesy (IBM i).

Kde je to možné, umístěte všechny struktury IBM MQ dohromady tak, aby byly všechny zarovnány podél hranic.

Kódování v jazyku assembler System/390 (rozhraní fronty zpráv)

Všimněte si informací v následujících sekcích při kódování programů IBM MQ for z/OS v jazyku assembler.

- “Řízení výpisu” na stránce 1032
- “Určení počátečních hodnot pro pole” na stránce 1032
- “Psaní znovu zapisovatelných programů” na stránce 1033
- “Použití CEDF” na stránce 1033

Názvy

Názvy parametrů v popisech volání a názvy polí v popisech struktur jsou zobrazeny ve smíšených případech. V makrech v assembleru dodaném s IBM MQjsou všechna jména psána velkými písmeny.

Použití volání MQI

Rozhraní MQI je rozhraní volání, takže programy v jazyku asembler musí dodržovat konvence sestavení operačního systému.

Zejména před tím, než vydá volání MQI, musí programy v jazyce asembleru odkazovat na registr R13 v oblasti pro ukládání s alespoň 18 úplnými slovy. Tato oblast pro ukládání dat poskytuje úložný prostor pro volaný program. Ukládá registry volajícího před zničením jejich obsahu a obnoví obsah registrů volajících při návratu.

Poznámka: To je důležité pro programy v jazyce Assembler CICS , které používají makro DFHEIENT k nastavení jejich dynamického úložiště, ale že se rozhodnete přepsat výchozí DATAREG z R13 na jiné registry. Když rozhraní správce Resource Manager produktu CICS přijímá řízení z stubu, uloží aktuální obsah registrů na adresu, na kterou ukazuje R13 . Selhávání při rezervaci oblasti pro ukládání za tímto účelem poskytuje nepředvídatelné výsledky a pravděpodobně způsobí nestandardní ukončení CICS.

Deklarace konstant

Většina konstant je deklarována jako rovnítko v makru CMQA.

Následující konstanty však nemohou být definovány jako rovnocenné a tyto konstanty nejsou zahrnuty, když zavoláte makro pomocí výchozích voleb:

- MQACT_NONE
- MQCI_NONE
- MQFMT_NONE
- MQFMT_ADMIN
- MQFMT_COMMAND_1
- MQFMT_COMMAND_2
- HLAVIČKA MQFMT_DEAD_LETTER_HEADER
- UDÁLOST MQFMT_EVENT
- MQFMT_IMS
- MQFMT_IMS_VAR_STRING
- MQFMT_PCF
- ŘETĚZEC MQFMT_STRING
- SPOUŠTĚČ MQFMT_TRIGGER
- ZÁHLAVÍ MQFMT_XMIT_Q_HEADER
- MQMI_NONE

Chcete-li je zahrnout, přidejte klíčové slovo EQUONLY=NE, když zavoláte makro.

CMQA je chráněn proti více deklaraci, takže jej můžete zahrnout mnohokrát. Klíčové slovo EQUEONLY však bude účinné pouze při prvním zahrnutí makra.

Zadání názvu struktury

Chcete-li povolit více než jednu instanci struktury, která má být deklarována, bude makro, které generuje strukturu, předpony názvu každého pole s řetězcem specifikovatelným uživatelem a podtržítkem (_).

Zadejte řetězec při vyvolání makra. Nezadáte-li řetězec, makro použije název struktury k sestavení předpony:

```
* Declare two object descriptors
CMQODA         Prefix used="MQOD_" (the default)
MY_MQOD CMQODA         Prefix used="MY_MQOD_"
```

Deklarace struktury v části Popisy volání zobrazují výchozí předponu.

Určení tvaru struktury

Makra mohou generovat deklarace struktury v jedné ze dvou tvarů, řízených parametrem DSECT:

DCET=ANO
Instrukce DSECT v asembleru-jazyk se použije ke spuštění nové sekce dat; definice struktury okamžitě následuje za příkazem DSECT. Není přiděleno žádné úložiště, takže inicializace není možná. Návěští pro vyvolání makra se použije jako název sekce dat; není-li zadán žádný popisek, použije se název struktury.

DSECT=NE
Instrukce DC assembler-language DC se používají k definování struktury na aktuální pozici v rutině. Pole jsou inicializována s hodnotami, které můžete zadat pomocí kódování relevantních parametrů při vyvolání makra. Pole, pro které nejsou zadány žádné hodnoty při vyvolání makra, jsou inicializovány výchozími hodnotami.

DSECT = NO se předpokládá, pokud parametr DSECT není zadán.

Řízení výpisu

Vzhled struktury deklarace struktury v asembleru-v seznamu můžete nastavit pomocí parametru LIST:

LIST=ANO
Deklarace struktury se zobrazí v seznamu v asembleru.

SEZNAM=NE
Deklarace struktury se neobjevuje ve výpisu v asembleru. Tento parametr se předpokládá, pokud není zadán parametr LIST.

Určení počátečních hodnot pro pole

Hodnotu, která má být použita k inicializaci pole ve struktuře, můžete zadat kódováním názvu tohoto pole (bez předpony) jako parametru při vyvolání makra spolu s požadovanou hodnotou.

Chcete-li například deklarovat strukturu deskriptoru zpráv s polem *MsgType* inicializovaným s hodnotou MQMT_REQUEST a pole *ReplyToQ* inicializováno s řetězcem MY_REPLT_TO_QUEUE, použijte následující kód:

```
MY_MQMD    CMQMDA      MSGTYPE=MQMT_REQUEST,       X
REPLYTOQ=MY_REPLY_TO_QUEUE
```

Uvedete-li pojmenovanou konstantu (nebo hodnotu equate) jako hodnotu při vyvolání makra, použijte makro CMQA k definování pojmenované konstanty. Nesmíte uzavřít do jednoduchých uvozovek ('') hodnoty, které jsou znakové řetězce.

Psaní znovu zapisovatelných programů

Produkt IBM MQ používá své struktury pro vstup i výstup. Pokud chcete, aby váš program zůstal znovu zadání:

1. Definujte verze pracovních úložišť struktur jako DSECT, nebo definujte struktury vložené v již definovaném DSECT. Pak zkopírujte DSECT do úložiště, které získáte pomocí:
   - Pro dávkové programy a programy TSO, makra STORAGE nebo GETMAIN z/OS assembler
   - Pro databázi CICS, pracovní úložný subsystém DSECT (DFHEISTG) nebo příkaz EXEC CICS GETMAIN

   Chcete-li správně inicializovat tyto pracovní struktury úložiště, zkopírujte konstantní verzi odpovídající struktury do pracovní verze úložiště.

   Poznámka: Struktury MQMD a MQXQH jsou o délce delší než 256 bajtů. Chcete-li zkopírovat tyto struktury do paměti, použijte instrukci assembleru MVCL.
2. Rezervní prostor v úložišti s použitím formuláře LIST (MF=L) makra CALL. Použijete-li makro CALL k provedení volání MQI, použijte formulář EXECUTE (MF=E) makra za použití dříve vyhrazeného úložiště, jak je uvedeno v příkladu pod volbou “Použití CEDF” na stránce 1033. Další příklady toho, jak to provést, naleznete v ukázkových programech v jazyce asembleru, které jsou dodávány s produktem IBM MQ.

Použijte volbu RENT jazyka v asembleru, která vám pomůže určit, zda je váš program znovu zadán.

Informace o zápisu znovuvydatelných programů najdete v příručce *z/OS MVS Application Development Guide: Assembler Language Programs*.

Použití CEDF

Chcete-li použít transakci dodanou CICS, CEDF (CICS Execution Diagnostic Facility), která vám pomůže ladit váš program, přidejte klíčové slovo ,VL ke každému příkazu CALL , například:

```
CALL MQCONN,(NAME,HCONN,COMPCODE,REASON),MF=(E,PARMAREA),VL
```

Předchozí příklad je reenterovatelný assembler-kód jazyka, kde PARMAREA je oblast v pracovním prostoru pro ukládání dat, kterou jste uvedli.

Použití volání MQI

Rozhraní MQI je rozhraní volání, takže programy v jazyku asembler musí dodržovat konvence sestavení operačního systému. Zejména před tím, než vydá volání MQI, musí programy v jazyce asembleru odkazovat na registr R13 v oblasti pro ukládání s alespoň 18 úplnými slovy. Tato oblast pro ukládání dat poskytuje úložný prostor pro volaný program. Ukládá registry volajícího před zničením jejich obsahu a obnoví obsah registrů volajících při návratu.

Poznámka: To je důležité pro programy v jazyce Assembler CICS , které používají makro DFHEIENT k nastavení jejich dynamického úložiště, ale že se rozhodnete přepsat výchozí DATAREG z R13 na jiné registry. Když rozhraní správce Resource Manager produktu CICS přijímá řízení z stubu, uloží aktuální obsah registrů na adresu, na kterou ukazuje R13 . Selhávání vhodné oblasti pro ukládání za tímto účelem poskytuje nepředvídatelné výsledky a pravděpodobně způsobí nestandardní ukončení práce systému CICS.

IBM i Kódování programů IBM MQ v jazyce RPG (pouzeIBM i)

V dokumentaci produktu IBM MQ jsou parametry volání, názvy datových typů, pole struktur a názvy konstant všechny popsány pomocí jejich dlouhých názvů. V RPG jsou tyto názvy zkráceny na šest nebo méně velkých písmen.

Například, pole *MsgType* se stane *MDMT* v RPG. Další informace naleznete v příručce IBM i Application Programming Reference (ILE/RPG).

Kódování v jazyce PL/I (pouze pro produktz/OS)

Užitečné informace při kódování pro IBM MQ v PL/I.

Struktury

Struktury jsou deklarovány s atributem ZALOŽENÝ, a tak nezabírají žádné úložiště, pokud program deklaruje jednu nebo více instancí struktury.

Instance struktury může být deklarována pomocí atributu `like` , například:

```
dcl my_mqmd    like MQMD; /* one instance */
dcl my_other_mqmd like MQMD; /* another one */
```

Pole struktury jsou deklarována s atributem INITIAL; je-li atribut `like` použit k deklaraci instance struktury, dědí tato instance počáteční hodnoty definované pro tuto strukturu. Je třeba nastavit pouze ta pole, kde se požadovaná hodnota liší od počáteční hodnoty.

Agent PL/I není citlivý na velká a malá písmena, takže názvy volání, polí struktury a konstant lze kódovat malými písmeny, velkými písmeny nebo se smíšenými malými a velkými písmeny.

Pojmenované konstanty

Pojmenované konstanty se deklarují jako proměnné makra; v důsledku toho se pojmenované konstanty, které nejsou odkazovány programem, nezabírají žádné úložiště v kompilovaném postupu.

Avšak volba kompilátoru, která způsobí, že se zdroj zpracuje preprocesorem makra, musí být uveden, když je program kompilován.

Všechny proměnné makra jsou znakové proměnné, dokonce i ty, které reprezentují číselné hodnoty. I když se to může zdát kontraintuitivní, nevyústí v žádný konflikt typu dat poté, co byly proměnné makra nahrazeny makroprocesorem, například:

```
%dcl MQMD_STRUC_ID char;
%MQMD_STRUC_ID = '''MD ''';

%dcl MQMD_VERSION_1 char;
%MQMD_VERSION_1 = '1';
```

Použití ukázkových procedurálních programů produktu IBM MQ

Tyto ukázkové programy jsou napsány v procedurálních jazycích a demonstrují typická použití rozhraní MQI (Message Queue Interface). Programy produktu IBM MQ na různých platformách.

Informace o této úloze

K dispozici jsou dvě sady ukázek:

- Ukázkové programy pro distribuované systémy a IBM i.
- Ukázkové programy pro produkt z/OS.

Procedura

- Chcete-li zjistit více o ukázkových programech, použijte následující odkazy:
  - “Použití ukázkových programů na více platformách” na stránce 1035
  - z/OS “Použití ukázkových programů pro produkt z/OS” na stránce 1137

Související pojmy

“Koncepty vývoje aplikací” na stránce 6

K zápisu aplikací IBM MQ můžete použít volbu procedurálních nebo objektově orientovaných jazyků. Odkazy v tomto tématu použijte pro informace o koncepcích produktu IBM MQ , které jsou užitečné pro vývojáře aplikací.

“Vyvíjení aplikací pro IBM MQ” na stránce 5
Můžete vyvíjet aplikace k odesílání a přijímání zpráv a ke správě správců front a souvisejících prostředků. IBM MQ podporuje aplikace napsané v mnoha různých jazycích a rámcích.

“Aspekty návrhu pro aplikace produktu IBM MQ” na stránce 43
Když se rozhodnete, jak vaše aplikace mohou využívat výhody platforem a prostředí, které máte k dispozici, musíte se rozhodnout, jak používat funkce nabízené produktem IBM MQ.

“Psaní procedurální aplikace pro řazení do fronty” na stránce 689
Prostřednictvím těchto informací můžete získat informace o vytváření aplikací pro řazení do front, připojování a odpojování od správce front, publikování a odběru a otevírání a zavírání objektů.

“Zápis procedurálních aplikací klienta” na stránce 874
Co potřebujete vědět, chcete-li psát klientské aplikace na serveru IBM MQ pomocí procedurálního jazyka.

“Zapisování aplikací typu publikování/odběr” na stránce 774
Začněte psát aplikace pro publikování/odběr aplikací produktu IBM MQ .

“Sestavení procedurální aplikace” na stránce 966
Aplikaci IBM MQ můžete psát v jednom z několika procedurálních jazyků a aplikaci spustit na několika různých platformách.

“Obsluha chyb procedurálních programů” na stránce 1015
Tyto informace vysvětlují chyby související s voláními MQI MQI buď při volání, nebo při doručení její zprávy do konečného cíle.

Související úlohy

“Vývoj webových služeb pomocí produktu IBM MQ” na stránce 1255
Můžete vyvíjet aplikace produktu IBM MQ pro webové služby pomocí přenosu IBM MQ pro protokol SOAP.

Multi Použití ukázkových programů na více platformách

Tyto vzorové procedurální programy se dodávají spolu s produktem. Ukázky jsou napsány v jazycích C a COBOL a demonstrují typická použití rozhraní MQI (Message Queue Interface).

Informace o této úloze

Ukázky nejsou určeny k demonstraci obecných programovacích technik, takže se vynechává některá kontrola chyb, kterou byste mohli chtít zahrnout do produkčního programu.

Zdrojový kód pro všechny ukázky je dodáván spolu s produktem; tento zdroj zahrnuje komentáře, které vysvětlují techniky front zpráv demonstrovány v programech.

IBM i Informace o programování v jazyce RPG najdete v příručce IBM i Application Programming Reference (ILE/RPG).

Názvy ukázek začínají předponou amq. Čtvrtý znak označuje programovací jazyk a kompilátor, je-li to nezbytné:

- s: Jazyk C
- 0: Jazyk COBOL na kompilátorech IBM i Micro Focus
- i: Jazyk COBOL pouze na kompilátorech IBM
- m: Jazyk COBOL pouze na kompilátorech Micro Focus

Osmý znak spustitelného souboru označuje, zda je ukázka spuštěna v režimu lokální vazby nebo v režimu klienta. Pokud žádný osmý znak neobsahuje, je ukázka spuštěna v režimu lokálních vazeb. Je-li osmý znak ' c', ukázka se spustí v režimu klienta.

Než budete moci spustit ukázkové aplikace, musíte nejprve vytvořit a nakonfigurovat správce front. Chcete-li nastavit správce front tak, aby přijímal klientská připojení, přečtěte si téma “Konfigurace správce front pro příjem klientských připojení na platformách Multiplatforms” na stránce 1045.

Procedura

- Chcete-li zjistit více o ukázkových programech, použijte následující odkazy:
  - “Vlastnosti demonstrovány v ukázkových programech na platformách Multiplatforms” na stránce 1036
  - “Ukázkové programy publikování/odběru” na stránce 1072
  - “Ukázkové programy Put” na stránce 1078
  - “Ukázkový program Distribuční seznam” na stránce 1065
  - “Ukázkové programy Procházet” na stránce 1053
  - “Ukázkový program prohlížeče” na stránce 1054
  - “Ukázkové programy Get” na stránce 1066
  - “Ukázkové programy Referenční zprávy” na stránce 1080
  - “Ukázkové programy požadavku” na stránce 1087
  - “Ukázkové programy pro zjišťování” na stránce 1071
  - “The Inquire Properties of a Message Handle sample program” na stránce 1072
  - “Ukázkové programy Set” na stránce 1092
  - “Ukázkové programy Echo” na stránce 1066
  - “Ukázkový program pro převod dat” na stránce 1057
  - “Spouštěcí ukázkové programy” na stránce 1096
  - “Ukázkový program Asynchronous Put” na stránce 1053
  - “Ukázky koordinace databází” na stránce 1058
  - “Ukázka transakce CICS” na stránce 1056
  - “Použití ukázek TUXEDO na systémech UNIX a Windows” na stránce 1097
  - “Ukázka obslužné rutiny fronty nedoručených zpráv” na stránce 1064
  - “Ukázkový program Connect” na stránce 1056
  - “Ukázkový program uživatelské procedury rozhraní API” na stránce 1051
  - “Použití uživatelské procedury zabezpečení SSPI v systému Windows” na stránce 1112
  - “Spuštění ukázek pomocí vzdálených front” na stránce 1113
  - “Ukázkový program pro monitorování front klastru (AMQSCLM)” na stránce 1113
  - “Ukázkový program pro vyhledávání koncového bodu připojení (CEPL)” na stránce 1123

Související pojmy

“Ukázkové programy C++” na stránce 488
K dispozici jsou čtyři vzorové programy, které demonstrují získávání a vkládání zpráv.

Související úlohy

“Použití ukázkových programů pro produkt z/OS” na stránce 1137
Ukázkové procedurální aplikace, které jsou dodávány s produktem IBM MQ for z/OS , demonstrují typická použití rozhraní MQI (Message Queue Interface).

Multi *Vlastnosti demonstrovány v ukázkových programech na platformách Multiplatforms*

Kolekce tabulek, které zobrazují techniky demonstrovány ukázkovými programy produktu IBM MQ .

Všechny ukázky otevírají a zavírají fronty pomocí volání MQOPEN a MQCLOSE, takže tyto techniky nejsou v tabulkách uvedeny odděleně. Podívejte se do záhlaví, které zahrnuje platformu, o kterou máte zájem.

z/OS Informace o platformě z/OS viz “Použití ukázkových programů pro produkt z/OS” na stránce 1137.

Linux UNIX *Ukázky pro systémy UNIX and Linux*

Techniky demonstrovány ukázkovými programy pro produkt IBM MQ v systémech UNIX and Linux .

Informace o tom, kde jsou uloženy ukázkové programy pro produkt IBM MQ v systémech UNIX and Linux , najdete v tématu “Příprava a spouštění ukázkových programů v systému UNIX and Linux” na stránce 1048 .

Tabulka 145 na stránce 1037 Tabulka uvádí, které zdrojové soubory C a COBOL jsou poskytovány, a zda je zahrnut server nebo klientský spustitelný soubor.

Tabulka 145. IBM MQ v ukázkových programech produktu UNIX and Linux demonstrujících použití rozhraní MQI (C a COBOL)

Technika	**C (zdroj) (“1” na stránce 1039)**	**COBOL (zdroj) (“2” na stránce 1039)**	**Server (spustitelný soubor C)**	**Klient (spustitelný soubor C) (“3” na stránce 1039)**
Použití rozhraní pro publikování/odběr	amqspuba amqssuba amqssbxa	bez vzorku	amqspub amqssub amqssbx	bez vzorku
Vložení zpráv pomocí volání MQPUT	amqsput0	amq0put0	amqsput	amqsputc
Vložení jedné zprávy pomocí volání MQPUT1	amqsinqa amqsecha	amqminqx amqmechx amqiinqx amqiechx	amqsinq amqsech	amqsechc
Vložení zpráv do rozdělovníku (“4” na stránce 1039)	amqsptl0	amq0ptl0.cbl	amqsptl	amqsptlc
Odpověď na zprávu požadavku	amqsinqa	amqminqx amqiinqx	amqsinq	bez vzorku
Získávání zpráv pomocí procházení (bez čekání)	amqsgbr0	amq0gbr0	amqsgbr	bez vzorku
Získávání zpráv (čekání s časovým limitem)	amqsget0	amq0get0	amqsget	amqsgetc
Získávání zpráv (neomezená doba čekání)	amqstrg0	bez vzorku	amqstrg	amqstrgc
Získávání zpráv (s převodem dat)	amqsecha	bez vzorku	amqsech	bez vzorku
Vložení odkazové zprávy do fronty (“4” na stránce 1039)	amqsprma	bez vzorku	amqsprm	amqsprc
Získání referenčních zpráv z fronty (“4” na stránce 1039)	amqsgrma	bez vzorku	amqsgrm	amqsgrmc
Referenční bod kanálu pro referenční zprávy (“4” na stránce 1039)	amqsqrma amqsxrma	bez vzorku	amqsxrm	bez vzorku
Procházení prvních 20 znaků zprávy	amqsgbr0	amq0gbr0	amqsgbr	amqsgbrc
Procházení kompletních zpráv	amqsbcg0	bez vzorku	amqsbcg	amqsbcgc

Tabulka 145. IBM MQ v ukázkových programech produktu UNIX and Linux demonstrujících použití rozhraní MQI (C a COBOL) (pokračování)

Technika	**C (zdroj) ("1" na stránce 1039)**	**COBOL (zdroj) ("2" na stránce 1039)**	**Server (spustitelný soubor C)**	**Klient (spustitelný soubor C) ("3" na stránce 1039)**
Použití sdílené vstupní fronty	amqsinqa	amqminqx amqiinqx	amqsinq	amqsinqc
Použití výlučné vstupní fronty	amqstrg0	amq0req0	amqstrg	amqstrgc
Použití volání MQINQ	amqsinqa	amqminqx amqiinqx	amqsinq	bez vzorku
Použití volání MQSET	amqsseta	amqmsetx amqisetx	amqsset	amqssetc
Použití fronty pro odpověď	amqsreq0	amq0req0	amqsreq	amqsreqc
Vyžádání výjimek zpráv	amqsreq0	amq0req0	amqsreq	bez vzorku
Přijímání oříznuté zprávy	amqsgbr0	amq0gbr0	amqsgbr	bez vzorku
Použití vyřešeného názvu fronty	amqsgbr0	amq0gbr0	amqsgbr	bez vzorku
Spouštění procesu	amqstrg0	bez vzorku	amqstrg	amqstrgc
Použití konverze dat	("5" na stránce 1039)	bez vzorku	bez vzorku	bez vzorku
IBM MQ (koordinující XA-kompatibilní správce databází) přístup k jedné databázi pomocí SQL	amqsxas0.sqc Db2 amqsxas0.ec Informix	amq0xas0.sqb	bez vzorku	bez vzorku
IBM MQ (koordinující XA-kompatibilní správce databází) přístup k dvěma databázím pomocí jazyka SQL	amqsxag0.c amqsxab0.sqc amqsxaf0.sqc	amq0xag0.cbl amq0xab0.sqb amq0xaf0.sqb	bez vzorku	bez vzorku
Transakce CICS ("6" na stránce 1039)	amqscic0.ccs	bez vzorku	amqscic0	bez vzorku
Encina transakce ("4" na stránce 1039)	amqsxae0	bez vzorku	amqsxae0	bez vzorku
Transakce TUXEDO pro vložení zpráv ("7" na stránce 1039)	amqstxpx	bez vzorku	bez vzorku	bez vzorku
Transakce TUXEDO pro získání zpráv ("7" na stránce 1039)	amqstxgx	bez vzorku	bez vzorku	bez vzorku
Server pro TUXEDO ("7" na stránce 1039)	amqstxsx	bez vzorku	bez vzorku	bez vzorku
obslužná rutina fronty nedoručených zpráv	Adresář `./tools/c/Samples/dlq` ("8" na stránce 1039)	bez vzorku	amqsdlq	bez vzorku

Tabulka 145. IBM MQ v ukázkových programech produktu UNIX and Linux demonstrujících použití rozhraní MQI (C a COBOL) (pokračování)

Technika	**C (zdroj) ("1" na stránce 1039)**	**COBOL (zdroj) ("2" na stránce 1039)**	**Server (spustitelný soubor C)**	**Klient (spustitelný soubor C) ("3" na stránce 1039)**
odeslání zprávy z klienta MQI	bez vzorku	bez vzorku	bez vzorku	amqsputc
Z klienta MQI se získáním zprávy	bez vzorku	bez vzorku	bez vzorku	amqsgetc
Připojení ke správci front pomocí příkazu MQCONNX	amqscnxc	bez vzorku	bez vzorku	amqscnxc
Použití ukončení rozhraní API	amqsaxe0	bez vzorku	amqsaxe	bez vzorku
Ukončení vyrovnávání pracovní zátěže klastru	amqswlm0	bez vzorku	amqswlm	bez vzorku
Asynchronně vkládání zpráv a získání stavu pomocí volání MQSTAT	amqsapt0	bez vzorku	amqsap	amqsaptc
Opakovaně připojitelní klienti	amqsphak amqsgak amqsmhak	bez vzorku	nelze použít	amqsphak amqsgak amqsmhak
Použití spotřebitelů zpráv k asynchronnímu příjmu zpráv z více front	amqscbf0	bez vzorku	amqscbf	amqscbfc
Zadání informací o připojení TLS pro MQCONNX	amqsssl c	bez vzorku	nelze použít	amqsssl c

Notes:

1. Spustitelná verze ukázek produktu IBM MQ MQI client sdílí stejný zdroj jako ukázky, které se spouštějí v prostředí serveru.
2. Kompilace programů začínajících 'amqm' se kompilátorem Micro Focus COBOL, ty začínající 'amqi' s kompilátorem jazyka IBM jazyka COBOL a s volbou 'amq0', které mají buď hodnotu.
3. Spustitelné verze ukázek produktu IBM MQ MQI client nejsou k dispozici v produktu IBM MQ for HP-UX.
4. Podporováno pouze na systémech IBM MQ for AIX, IBM MQ for HP-UXa IBM MQ for Solaris .
5. V systémech IBM MQ for AIX, IBM MQ for HP-UXa IBM MQ for Solaris se tento program nazývá `amqsvfc0.c` .
6. Produkt CICS je podporován pouze produkty IBM MQ for AIX a IBM MQ for HP-UX .
7. Produkt TUXEDO není podporován produktem IBM MQ for Linux v systému System p.
8. Zdroj pro popisovač fronty nedoručených zpráv se skládá z několika souborů a je uveden v samostatném adresáři.

Chcete-li získat podrobné informace o podpoře systémů UNIX and Linux , prohlédněte si téma Systémové požadavky pro IBM MQ.

Windows *Ukázky pro IBM MQ for Windows*

Techniky demonstrovány ukázkovými programy pro IBM MQ for Windows.

Seznamy Tabulka 146 na stránce 1040 uvádějí, které zdrojové soubory C a COBOL jsou k dispozici a zda je zahrnut server nebo klientský spustitelný soubor.

Tabulka 146. Ukázkové programy produktu IBM MQ for Windows demonstrující použití rozhraní MQI (C a COBOL)

Technika	C (zdroj)	COBOL (zdroj)	Server (spustitelný soubor C)	Klient (spustitelný soubor C)
Použití rozhraní pro publikování/odběr	amqspuba amqssuba amqssbxa	bez vzorku	amqspub amqssub amqssbx	bez vzorku
Vložení zpráv pomocí volání MQPUT	amqsput0	amq0put0	amqsput	amqsputc
Vložení jedné zprávy pomocí volání MQPUT1	amqsinqa amqsecha	amqminq2 amqmech2 amqiinq2 amqiech2	amqsinq amqsech	amqsinqc amqsechc
Vložení zpráv do distribučního seznamu	amqsptl0	amq0ptl0.cbl	amqsptl	amqsptlc
Odpověď na zprávu požadavku	amqsinqa	amqminq2 amqiinq2	amqsinq	amqsinqc
Získávání zpráv (bez čekání)	amqsgbr0	amq0gbr0	amqsgbr	amqsgbrc
Získávání zpráv (čekání s časovým limitem)	amqsget0	amq0get0	amqsget	amqsgetc
Získávání zpráv (neomezená doba čekání)	amqstrg0	bez vzorku	amqstrg	amqstrgc
Získávání zpráv (s převodem dat)	amqsecha	bez vzorku	amqsech	amqsechc
Vložení odkazové zprávy do fronty	amqsprma	bez vzorku	amqsprm	amqsprc
Získání referenčních zpráv z fronty	amqsgrma	bez vzorku	amqsgrm	amqsgrmc
Referenční uživatelská procedura kanálu zpráv	amqsqrma amqsxrma	bez vzorku	amqsxrm	bez vzorku
Procházení prvních 20 znaků zprávy	amqsgbr0	amq0gbr0	amqsgbr	amqsgbrc
Procházení kompletních zpráv	amqsbcg0	bez vzorku	amqsbcg	amqsbcgc
Použití sdílené vstupní fronty	amqsinqa	amqminq2 amqiinq2	amqsinq	amqsinqc
Použití výlučné vstupní fronty	amqstrg0	amq0req0	amqstrg	amqstrgc
Použití volání MQINQ	amqsinqa	amqminq2 amqiinq2	amqsinq	amqsinqc
Použití volání MQSET	amqsseta	amqmset2 amqiset2	amqsset	amqssetc
Použití volání MQINQMP	amqsiqma	bez vzorku	bez vzorku	bez vzorku
Použití fronty pro odpověď	amqsreq0	amq0req0	amqsreq	amqsreqc
Vyžádání výjimek zpráv	amqsreq0	amq0req0	amqsreq	amqsreqc
Přijímání ořízntuté zprávy	amqsgbr0	amq0gbr0	amqsgbr	amqsgbrc
Použití vyřešeného názvu fronty	amqsgbr0	amq0gbr0	amqsgbr	amqsgbrc
Spouštění procesu	amqstrg0	bez vzorku	amqstrg	amqstrgc
Použití konverze dat	amqsvfc0	bez vzorku	bez vzorku	bez vzorku

Tabulka 146. Ukázkové programy produktu IBM MQ for Windows demonstrující použití rozhraní MQI (C a COBOL) (pokračování)

Technika	**C (zdroj)**	**COBOL (zdroj)**	**Server (spustitelný soubor C)**	**Klient (spustitelný soubor C)**
IBM MQ (koordinující XA-kompatibilní správce databází) přístup k jedné databázi pomocí SQL	amqsxas0.sqc Db2 amqsxas0.ec Informix	amq0xas0.sqb	bez vzorku	bez vzorku
IBM MQ (koordinující XA-kompatibilní správce databází) přístup k dvěma databázím pomocí jazyka SQL	amqsxag0.c amqsxab0.sqc Db2 amqsxaf0.sqc Db2	amq0xag0.cbl amq0xab0.sqb amq0xaf0.sqb	bez vzorku	bez vzorku
Transakce TUXEDO pro vložení zpráv	amqstxpx	bez vzorku	bez vzorku	bez vzorku
Transakce TUXEDO pro získání zpráv	amqstxgx	bez vzorku	bez vzorku	bez vzorku
Server pro TUXEDO	amqstxsx	bez vzorku	bez vzorku	bez vzorku
obslužná rutina fronty nedoručených zpráv	Adresář `./tools/c/Samples/dlq` (“1” na stránce 1042)	bez vzorku	amqsdlq	bez vzorku
Z IBM MQ MQI client, vložení zprávy	bez vzorku	bez vzorku	bez vzorku	amqsputc
Z IBM MQ MQI clientza účelem získání zprávy	bez vzorku	bez vzorku	bez vzorku	amqsgetc
Připojení ke správci front pomocí příkazu MQCONNX	amqscnxc	bez vzorku	bez vzorku	amqscnxc
Použití ukončení rozhraní API	amqsaxe0	bez vzorku	amqsaxe	bez vzorku
Vyrovnávání zátěže klastru	amqswlm0	bez vzorku	amqswlm	bez vzorku
Rutiny zabezpečení SSPI	amqsspin	bez vzorku	amqrspin.dll	amqrspin.dll
Asynchronně vkládání zpráv a získání stavu pomocí volání MQSTAT	amqsapt0	bez vzorku	amqsap	amqsaptc
Opakovaně připojitelní klienti	amqsphak amqsgak amqsmhak	bez vzorku	Nelze použít	amqsphak amqsgak amqsmhak
Použití spotřebitelů zpráv k asynchronnímu příjmu zpráv z více front	amqscbf0	bez vzorku	amqscbf	amqscbfc
Zadání informací o připojení TLS pro MQCONNX	amqssslc	bez vzorku	nelze použít	amqssslc

Notes:

1. Zdroj pro popisovač fronty nedoručených zpráv se skládá z několika souborů a je uveden v samostatném adresáři.

Windows *Ukázky jazyka Visual Basic pro produkt IBM MQ for Windows*

Techniky demonstrovány ukázkovými programy pro produkt IBM MQ v systémech Windows .

Tabulka 147 na stránce 1042 zobrazuje techniky demonstrovány ukázkovými programy produktu IBM MQ for Windows .

Projekt může obsahovat několik souborů. Když otevřete projekt v rámci Visual Basic, ostatní soubory se načtou automaticky. Nejsou k dispozici žádné spustitelné programy.

Všechny vzorové projekty, kromě mqtrivc.vbp, jsou nastaveny pro práci se serverem IBM MQ . Informace o tom, jak lze změnit ukázkové projekty pro práci s klienty produktu IBM MQ , naleznete v tématu “Příprava programů produktu Visual Basic v produktu Windows” na stránce 998.

Tabulka 147. Ukázkové programy produktu IBM MQ for Windows demonstrující použití rozhraní MQI (Visual Basic)

Technika	**Název souboru projektu**
Vložení zpráv pomocí volání MQPUT	amqsputb.vbp
Získávání zpráv pomocí volání MQGET	amqsgetb.vbp
Procházení fronty pomocí volání MQGET	amqsbcgb.vbp
Jednoduchá MQGET a ukázka MQPUT (klient)	mqtrivc.vbp
Jednoduchá MQGET a ukázka MQPUT (server)	mqtrivs.vbp
Vložení a získání řetězců a uživatelem definovaných struktur pomocí MQPUT a MQGET	strings.vbp
Použití struktur PCF ke spuštění a zastavení kanálu	pcfsamp.vbp
Vytvoření fronty s použitím rozhraní MQAI	amqsaicq.vbp
Zobrazení seznamu front správce front pomocí rozhraní MQAI	amqsailq.vbp
Monitorování událostí pomocí rozhraní MQAI	amqsaiem.vbp

IBM i *Ukázky pro IBM i*

Techniky demonstrovány ukázkovými programy pro produkt IBM MQ v systémech IBM i .

Tabulka 148 na stránce 1042 zobrazuje techniky demonstrovány ukázkovými programy produktu IBM MQ for IBM i . Některé techniky se vyskytují ve více než jednom ukázkovém programu, ale v tabulce je uveden pouze jeden program.

Tabulka 148. Ukázkové programy demonstrující použití rozhraní MQI (C a COBOL) v systému IBM i

Technika	**C (zdroj) (“1” na stránce 1044)**	**COBOL (zdroj) (“2” na stránce 1044)**	**RPG (zdroj) (“3” na stránce 1044)**	**Klient (spustitelný soubor C) (4)**
Vložení zpráv pomocí volání MQPUT	AMQSPUT0	AMQ0PUT4	AMQ3PUT4	amqsputc
Vložení zpráv z datového souboru pomocí volání MQPUT	AMQSPUT4	bez vzorku	bez vzorku	bez vzorku
Vložení jedné zprávy pomocí volání MQPUT1	AMQSINQ4, AMQSECH4	AMQ0INQ4, AMQ0ECH4	AMQ3INQ4, AMQ3ECH4	AMQSINQC, AMQSECHC
Vložení zpráv do distribučního seznamu	AMQSPTL4	bez vzorku	bez vzorku	AMQSPTLCK

Tabulka 148. Ukázkové programy demonstrující použití rozhraní MQI (C a COBOL) v systému IBM i (pokračování)

Technika	**C (zdroj) ("1" na stránce 1044)**	**COBOL (zdroj) ("2" na stránce 1044)**	**RPG (zdroj) ("3" na stránce 1044)**	**Klient (spustitelný soubor C) (4)**
Odpověď na zprávu požadavku	AMQSINQ4	AMQ0INQ4	AMQ3INQ4	AMQSINQC
Získávání zpráv (bez čekání)	AMQSGBR4	AMQ0GBR4	AMQ3GBR4	AMQSGBRC
Získávání zpráv (čekání s časovým limitem)	AMQSGET4	AMQ0GET4	AMQ3GET4	AMQSGETC.
Získávání zpráv (neomezená doba čekání)	AMQSTRG4	bez vzorku	AMQ3TRG4	AMQSTRGC
Získávání zpráv (s převodem dat)	AMQSECH4	AMQ0ECH4	AMQ3ECH4	AMQSECHC
Vložení odkazové zprávy do fronty	AMQSPRM4	bez vzorku	bez vzorku	AMQSPRC
Získání referenčních zpráv z fronty	AMQSGRM4	bez vzorku	bez vzorku	AMQGRMC
Referenční uživatelská procedura kanálu zpráv	AMQSQRM4, AMQSXRM4	bez vzorku	bez vzorku	žádná ukázka
Ukončení zprávy	AMQSCMX4	bez vzorku	bez vzorku	žádná ukázka
Procházení prvních 49 znaků zprávy	AMQSGBR4	AMQ0GBR4	AMQ3GBR4	AMQSGBRC
Procházení kompletních zpráv	AMQSBCG4	bez vzorku	bez vzorku	AMQSBCGC
Použití sdílené vstupní fronty	AMQSINQ4	AMQ0INQ4	AMQ3INQ4	AMQSINQC
Použití výlučné vstupní fronty	AMQSREQ4	AMQ0REQ4	AMQ3REQ4	AMQSREQCUS
Použití volání MQINQ	AMQSINQ4	AMQ0INQ4	AMQ3INQ4	AMQSINQC
Použití volání MQSET	AMQSSET4	AMQ0SET4	AMQ3SET4	AMQSSETC
Použití fronty pro odpověď	AMQSREQ4	AMQ0REQ4	AMQ3REQ4	AMQSREQCUS
Vyžádání výjimek zpráv	AMQSREQ4	AMQ0REQ4	AMQ3REQ4	AMQSREQCUS
Přijímání ořízntuté zprávy	AMQSGBR4	AMQ0GBR4	AMQ3GBR4	AMQSGBRC
Použití vyřešeného názvu fronty	AMQSGBR4	AMQ0GBR4	AMQ3GBR4	AMQSGBRC
Spouštění procesu	AMQSTRG4	bez vzorku	AMQ3TRG4	AMQSTRGC
Spouštěcí server	AMQSERV4	bez vzorku	AMQ3SRV4	bez vzorku
Použití spouštěcího serveru (včetně transakcí CICS)	AMQSERV4	bez vzorku	AMQ3SRV4	bez vzorku
Použití konverze dat	AMQSVFC4	bez vzorku	bez vzorku	bez vzorku
Použití ukončení rozhraní API	AMQSAXE0	bez vzorku	bez vzorku	bez vzorku
Vyrovnávání zátěže klastru	AMQSWLM0	bez vzorku	bez vzorku	bez vzorku
Asynchronně vkládání zpráv a získání stavu pomocí volání MQSTAT	AMQSAPT0	bez vzorku	bez vzorku	AMQSAPTC
Použití rozhraní pro publikování/odběr	AMQSPUBA, AMQSSUBA, AMQSSBXA	bez vzorku	bez vzorku	AMQSPUBC, AMQSSUBC, AMQSSBXC

Tabulka 148. Ukázkové programy demonstrující použití rozhraní MQI (C a COBOL) v systému IBM i (pokračování)				
Technika	**C (zdroj) ("1" na stránce 1044)**	**COBOL (zdroj) ("2" na stránce 1044)**	**RPG (zdroj) ("3" na stránce 1044)**	**Klient (spustitelný soubor C) (4)**
Znovu připojitelné klienty (5)	AMQSPHAC, AMQSGHAC, AMQSMHAC,	bez vzorku	bez vzorku	bez vzorku
Použití spotřebitelů zpráv k asynchronnímu příjmu zpráv z více front (5)	AMQSCBFO	bez vzorku	bez vzorku	bez vzorku
Zadání informací o připojení TLS pro MQCONNX	AMQSSSLC	bez vzorku	bez vzorku	AMQSSSLC
Připojení ke správci front pomocí příkazu MQCONNX	AMQSCNXC	bez vzorku	bez vzorku	AMQSCNXC

Notes:

1. Zdroj pro ukázky jazyka C se nachází v souboru QMQMSAMP/QCSRC. Soubory začlenění se nacházejí jako členové v souboru QMQM/H.
2. Zdroj pro ukázky jazyka COBOL je v souborech QMQMSAMP/QCBLLESRC. Členové jsou pojmenovány AMQ0 xxx 4, kde xxx označuje ukázkovou funkci.
3. Zdroj pro ukázky RPG je v QMQMSAMP/QRPGLESRC. Členové jsou pojmenovány AMQ3 xxx 4, kde xxx označuje ukázkovou funkci. Kopírování členů existuje v QMQM/QRPGLESRC. Každý název člena má příponu G.
4. Spustitelná verze ukázek produktu IBM MQ MQI client sdílí stejný zdroj jako ukázky, které se spouštějí v prostředí serveru. Zdroj pro ukázky v prostředí klienta je stejný jako server. Ukázky IBM MQ MQI client jsou propojeny s knihovnou klienta LIBMQIC a IBM MQ ukázky serveru jsou propojeny s knihovnou serveru LIBMQM.
5. Pokud má být spuštěn spustitelný soubor klienta pro ukázkovou aplikaci klienta opětovného připojení a asynchronní aplikace spotřebitele, musí být kompilován a propojen s knihovnou vláken LIBMQIC_R. Proto musí být spuštěn ve vláknovém prostředí. Nastavte proměnnou prostředí QIBM_MULTI_THREADED na hodnotu 'Y' a spusťte aplikaci z qsh.

   Další informace naleznete v tématu Nastavení produktu IBM MQ pomocí jazyka Java a platformy JMS .

Kromě těchto voleb obsahuje ukázková volba IBM MQ for IBM i ukázkový datový soubor, který používáte jako vstup do ukázkových programů, AMQSDATA a ukázkové programy CL, které demonstrují administrativní úlohy. Ukázky CL jsou popsány v části Administrace produktu IBM i . Vzorový CL program amqsamp4 můžete použít k vytvoření front, které se mají použít s ukázkovými programy popsanými v tomto tématu.

Windows IBM i UNIX

Příprava a spuštění ukázkových programů

Po dokončení přípravy úvodní přípravy můžete ukázkové programy spustit.

Informace o této úloze

Před spuštěním ukázkových programů je třeba nejprve vytvořit správce front a vytvořit také fronty, které potřebujete. Možná budete také muset provést nějakou další přípravu, například, chcete-li spustit ukázky COBOL. Po dokončení nezbytné přípravy můžete potom spustit ukázkové programy.

Postup

Informace o tom, jak připravit a spustit ukázkové programy, najdete v následujících tématech:

- "Příprava a spouštění ukázkových programů v systému IBM i" na stránce 1047

- “Příprava a spouštění ukázkových programů v systému UNIX and Linux” na stránce 1048
- “Příprava a spouštění ukázkových programů v systému Windows” na stránce 1049

Konfigurace správce front pro příjem klientských připojení na platformách Multiplatforms

Než budete moci ukázkové aplikace spustit, je třeba nejprve vytvořit správce front. Pak můžete správce front nakonfigurovat tak, aby bezpečně přijímal příchozí požadavky na připojení od aplikací spuštěných v režimu klienta.

Než začnete

Ujistěte se, že správce front již existuje a že byl spuštěn. Určete, zda jsou již záznamy ověřování kanálu povoleny, zadáním příkazu MQSC:

```
DISPLAY QMGR CHLAUTH
```

Důležité: Tato úloha očekává, že jsou povoleny záznamy ověření kanálu. Pokud se jedná o správce front používaný ostatními uživateli a aplikacemi, změna tohoto nastavení bude mít vliv na všechny ostatní uživatele a aplikace. Pokud váš správce front nevyužívá záznamy ověření kanálu, může být krok 4 nahrazen alternativní metodou ověření (například uživatelská procedura zabezpečení), která nastaví MCAUSER na *neprivilegované-ID-uživatele* , které získáte v kroku “1” na stránce 1045.

Musíte vědět, jaký název kanálu očekává vaše aplikace, aby mohla být aplikace povolena pro použití kanálu. Je třeba také vědět, které objekty, například fronty nebo témata, očekává, že aplikace bude používat aplikaci tak, aby je bylo možné používat.

Informace o této úloze

Tato úloha vytvoří neprivilegované ID uživatele, které má být použito pro aplikaci klienta, která se připojuje ke správci front. Přístup pro klientskou aplikaci je udělen pouze k tomu, aby mohl používat kanál, který potřebuje, a frontu, kterou potřebuje k použití tohoto ID uživatele.

Postup

1. Získejte ID uživatele na systému, na kterém je spuštěný správce front. Pro tuto úlohu nesmí být toto ID uživatele privilegovaný administrativní uživatel. Toto ID uživatele bude oprávněním, pod kterým bude spuštěno připojení klienta ve správci front.
2. Spusťte program listener s následujícími příkazy, kde:

   qmgr-name je název vašeho správce front
   nnnn je číslo vybraného portu

   a) 

   Pro systémy UNIX a Windows :

   ```
   runmqlsr -t tcp -m qmgr-name -p nnnn
   ```

   b) 

   Pro produkt IBM i:

   ```
   STRMQMLSR MQMNAME(qmgr-name) PORT(nnnn)
   ```

3. Používá-li aplikace SYSTEM.DEF.SVRCONN je tento kanál již definován. Pokud vaše aplikace používá jiný kanál, vytvořte jej zadáním příkazu MQSC:

   ```
   DEFINE CHANNEL(' channel-name ') CHLTYPE(SVRCONN) TRPTYPE(TCP) +
   DESCR('Channel for use by sample programs')
   ```

název-kanálu je název vašeho kanálu.

4. Vytvořte pravidlo pro ověření kanálu, které umožňuje použití kanálu zadáním příkazu MQSC pouze zadáním adresy IP vašeho klienta:

```
SET CHLAUTH(' channel-name ') TYPE(ADDRESSMAP) ADDRESS(' client-machine-IP-address ') +
MCAUSER(' non-privileged-user-id ')
```

název-kanálu je název vašeho kanálu.

client-machine-IP-address je adresa IP vašeho klientského systému.

Je-li ukázková aplikace klienta spuštěna na stejném počítači jako správce front, použijte adresu IP '127.0.0.1', pokud se vaše aplikace chystá připojit pomocí 'localhost'. Pokud se chystáte připojit několik různých klientských počítačů, můžete použít vzor nebo rozsah místo jedné IP adresy. Podrobnosti viz Generické adresy IP .

non-privileged-user-id je ID uživatele, které jste získali v kroku “1” na stránce 1045

5. Používá-li aplikace SYSTEM.DEFAULT.LOCAL.QUEUE je tato fronta již definována. Pokud vaše aplikace používá jinou frontu, vytvořte ji zadáním příkazu MQSC:

```
DEFINE QLOCAL(' queue-name ') DESCR('Queue for use by sample programs')
```

queue-name je název vaší fronty.

6. Udělte přístup pro připojení k správci front a dotazujte jej:

a) Windows IBM i UNIX

Pro systémy IBM i IBM i, UNIX a Windows zadejte příkazy MQSC:

```
SET AUTHREC OBJTYPE(QMGR) PRINCIPAL(' non-privileged-user-id ') +
AUTHADD(CONNECT, INQ)
```

non-privileged-user-id je ID uživatele, které jste získali v kroku “1” na stránce 1045

7. Je-li vaše aplikace aplikací typu point-to-point, znamená to použití front, udělení přístupu k povolení a odesílání a získání zpráv pomocí vaší fronty pomocí ID uživatele, který má být použit, vydáním příkazů MQSC:

a) Windows IBM i UNIX

Pro systémy IBM i IBM i, UNIX a Windows zadejte příkazy MQSC:

```
SET AUTHREC PROFILE(' queue-name ') OBJTYPE(QUEUE) +
PRINCIPAL(' non-privileged-user-id ') AUTHADD(PUT, GET, INQ, BROWSE)
```

queue-name je název vaší fronty.

non-privileged-user-id je ID uživatele, které jste získali v kroku “1” na stránce 1045

8. Je-li vaše aplikace aplikací typu publikování-odběr, která využívá témata, udělte přístup k povolení publikování a přihlášení k odběru tématu pomocí ID uživatele, který má být použit, zadáním příkazů MQSC:

a) Windows IBM i UNIX

Pro systémy IBM i IBM i, UNIX a Windows zadejte příkazy MQSC:

```
SET AUTHREC PROFILE('SYSTEM.BASE.TOPIC') OBJTYPE(TOPIC) +
PRINCIPAL(' non-privileged-user-id ') AUTHADD(PUB, SUB)
```

non-privileged-user-id je ID uživatele, které jste získali v kroku “1” na stránce 1045

Tato akce poskytne uživateli *neprivilegované-ID-uživatele* přístup k libovolnému tématu ve stromu témat, případně můžete definovat objekt tématu pomocí produktu **DEFINE TOPIC**

a udělit přístup pouze k části stromu témat odkazovaného daným objektem tématu. Podrobnosti najdete v tématu Řízení přístupu uživatelů k tématům .

Jak pokračovat dále

Klientská aplikace se nyní může připojit ke správci front a odesílat nebo přijímat zprávy s použitím fronty.

Související informace

SET CHLAUTH

Definovat kanál

DEFINOVAT QLOCAL

SET AUTHREC

IBM i Oprávnění IBM MQ v systému IBM i

Poskytnutí přístupu k objektu IBM MQ v systémech UNIX nebo Linux a Windows

IBM i *Příprava a spouštění ukázkových programů v systému IBM i*

Před spuštěním ukázkových programů v systému IBM ije třeba nejprve vytvořit správce front a také vytvořit fronty, které potřebujete. Pokud chcete spouštět ukázky jazyka COBOL, možná budete muset provést další přípravu.

Informace o této úloze

Zdroj pro ukázkové programy produktu IBM MQ for IBM i je poskytován v knihovně QMQMSAMP jako členy QCSRC, QCLSRC, QCBLLESRC a QRPGLESRC.

Při spouštění ukázek můžete použít vlastní fronty nebo můžete spustit ukázkový program AMQSAMP4 a vytvořit některé ukázkové fronty. Zdroj pro program AMQSAMP4 je obsažen v souboru QCLSRC v knihovně QMQMSAMP. Můžete ji zkompilovat pomocí příkazu CRTCLPGM.

Chcete-li ukázky spustit, použijte spustitelné verze jazyka C, které jsou dodány v knihovně QMQM, nebo je kompilujte podobným způsobem jako jakákoli jiná aplikace produktu IBM MQ .

Postup

1. Vytvořte správce front a nastavte výchozí definice.

   Než budete moci spustit některý z ukázkových programů, musíte to provést. Další informace o vytváření správce front naleznete v tématu Administrace produktu IBM MQ. Informace o konfiguraci správce front tak, aby bezpečně přijímal požadavky na příchozí připojení od aplikací spuštěných v režimu klienta, viz “Konfigurace správce front pro příjem klientských připojení na platformách Multiplatforms” na stránce 1045.

2. Chcete-li volat jeden z ukázkových programů pomocí dat z členu PUT v souboru AMQSDATA knihovny QMQMSAMP, použijte příkaz podobný tomuto:

   ```
   CALL PGM(QMQM/AMQSPUT4) PARM('QMQMSAMP/AMQSDATA(PUT)')
   ```

   Poznámka: Chcete-li pro kompilovaný modul použít systém souborů IFS, uveďte volbu SYSIFCOPT (*IFSIO) na CRTCMMOD, pak název souboru, předaný jako parametr, musí být uveden v následujícím formátu:

   ```
   home/me/myfile
   ```

3. Chcete-li použít příklady jazyka COBOL pro příklady Inquire, Set a Echo, změňte definice procesu před spuštěním těchto ukázek.

   Pro příklady Inquire, Set a Echo spustí ukázkové definice verze C těchto ukázek. Chcete-li používat verze jazyka COBOL, musíte změnit definice procesu:

   - SYSTEM.SAMPLE.INQPROCESS

- SYSTEM.SAMPLE.SETPROCESS
- SYSTEM.SAMPLE.ECHOPROCESS

V systému IBM imůžete použít příkaz **CHGMQMPRC** (podrobnosti naleznete v tématu Změna procesu MQ (CHGMQMPRC)) nebo upravit a spustit příkaz **AMQSAMP4** s alternativní definicí.

4. Spusťte ukázkové programy.

   Další informace o parametrech, které každý z ukázek očekává, najdete v popisech jednotlivých ukázek.

   Poznámka: U ukázkových programů v jazyce COBOL při předávání názvů front jako parametrů je třeba zadat 48 znaků a v případě potřeby vyplňující prázdné znaky. Cokoli jiného než 48 znaků způsobí, že program selže s kódem příčiny 2085.

Související odkazy

“Ukázky pro IBM i” na stránce 1042
Techniky demonstrovány ukázkovými programy pro produkt IBM MQ v systémech IBM i .

UNIX *Příprava a spouštění ukázkových programů v systému UNIX and Linux*

Před spuštěním ukázkových programů v systému UNIXje třeba nejprve vytvořit správce front a také vytvořit fronty, které potřebujete. Pokud chcete spouštět ukázky jazyka COBOL, možná budete muset provést další přípravu.

Informace o této úloze

Vzorové soubory IBM MQ na systémech UNIX and Linux se nacházejí v adresářích uvedených v parametru Tabulka 149 na stránce 1048 , pokud byly při instalaci použity výchozí hodnoty.

Tabulka 149. Kde najít ukázky pro IBM MQ v systémech UNIX and Linux

Obsah	**Adresář**
Zdrojové soubory	*MQ_INSTALLATION_PATH*/samp
zdrojové soubory obslužné rutiny fronty nedoručených zpráv	*MQ_INSTALLATION_PATH*/samp/dlq
spustitelné soubory	*MQ_INSTALLATION_PATH*/samp/bin

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Ukázky potřebují sadu front, se kterými se bude pracovat. K vytvoření sady můžete použít buď vlastní fronty, nebo spustit ukázkový soubor MQSC amqscos0.tst . Chcete-li ukázky spustit, buď použijte spustitelné verze dodané, nebo zkompilujte zdrojové verze jako jakékoli jiné aplikace, a to pomocí kompilátoru ANSI.

Postup

1. Vytvořte správce front a nastavte výchozí definice.

   Než budete moci spustit některý z ukázkových programů, musíte to provést. Další informace o vytváření správce front naleznete v tématu Administrace produktu IBM MQ. Informace o konfiguraci správce front tak, aby bezpečně přijímal požadavky na příchozí připojení od aplikací spuštěných v režimu klienta, viz “Konfigurace správce front pro příjem klientských připojení na platformách Multiplatforms” na stránce 1045.
2. Pokud nepoužíváte své vlastní fronty, spusťte ukázkový soubor MQSC amqscos0.tst pro vytvoření sady front.

   Chcete-li to provést na systémech UNIX and Linux , zadejte:

   ```
   runmqsc QManagerName <amqscos0.tst > /tmp/sampobj.out
   ```

   Zkontrolujte soubor `sampobj.out` , abyste se ujistili, že se nevyskytly žádné chyby.

3. Chcete-li použít příklady jazyka COBOL pro příklady Inquire, Set a Echo, změňte definice procesu před spuštěním těchto ukázek.

   Pro příklady Inquire, Set a Echo spustí ukázkové definice verze C těchto ukázek. Chcete-li používat verze jazyka COBOL, musíte změnit definice procesu:

   - SYSTEM.SAMPLE.INQPROCESS
   - SYSTEM.SAMPLE.SETPROCESS
   - SYSTEM.SAMPLE.ECHOPROCESS

   V systému Windowsto provedete tak, že upravíte soubor `amqscos0.tst` a změníte názvy spustitelných souborů jazyka C na názvy spustitelných souborů v jazyce COBOL dříve, než použijete příkaz **runmqsc** ke spuštění těchto ukázek.
4. Spusťte ukázkové programy.

   Chcete-li spustit ukázku, zadejte její název následovaný libovolkovými parametry, například:

   ```
   amqsput myqueue qmanagername
   ```

   kde *myqueue* je název fronty, na kterou se mají zprávy vložit, a *qmanagername* je správce front, který vlastní *myqueue*.

   Další informace o parametrech, které každý z ukázek očekává, najdete v popisech jednotlivých ukázek.

   Poznámka: U ukázkových programů v jazyce COBOL při předávání názvů front jako parametrů je třeba zadat 48 znaků a v případě potřeby vyplňující prázdné znaky. Cokoli jiného než 48 znaků způsobí, že program selže s kódem příčiny 2085.

Související odkazy

“Ukázky pro systémy UNIX and Linux” na stránce 1037
Techniky demonstrovány ukázkovými programy pro produkt IBM MQ v systémech UNIX and Linux .

Windows *Příprava a spouštění ukázkových programů v systému Windows*

Před spuštěním ukázkových programů v systému Windowsje třeba nejprve vytvořit správce front a také vytvořit fronty, které potřebujete. Pokud chcete spouštět ukázky jazyka COBOL, možná budete muset provést další přípravu.

Informace o této úloze

Ukázkové soubory produktu IBM MQ for Windows se nacházejí v adresářích uvedených v parametru Tabulka 150 na stránce 1049, pokud byly při instalaci použity výchozí hodnoty. Výchozí nastavení jednotky instalace je < c: >.

Tabulka 150. Kde najít ukázky pro IBM MQ for Windows

Obsah	**Adresář**
Zdrojový kód C	*MQ_INSTALLATION_PATH*\Tools\C\Samples
Zdrojový kód pro ukázku obslužné rutiny dead-letter	*MQ_INSTALLATION_PATH*\Tools\C\Samples\DLQ
Zdrojový kód COBOL	*MQ_INSTALLATION_PATH*\Tools\Cobol \ Samples
Spustitelné soubory C [1]	*MQ_INSTALLATION_PATH*\ Tools\C\Samples \ Bin (32bitové verze) *MQ_INSTALLATION_PATH*\ Tools\C\Samples\Bin64 (64bitové verze)
Ukázkové soubory MQSC	*MQ_INSTALLATION_PATH*\Tools\MQSC\Samples
Zdrojový kód Visual Basic	*MQ_INSTALLATION_PATH*\Tools\VB\SampVB6
Ukázky produktu .NET	*MQ_INSTALLATION_PATH*\Tools\dotnet \ Samples

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Poznámka: 64bitové verze jsou k dispozici pro některé ukázky spustitelných souborů jazyka C.

Ukázky potřebují sadu front, se kterými se bude pracovat. K vytvoření sady front můžete buď použít vlastní fronty, nebo spustit ukázkový soubor MQSC amqscos0.tst . Chcete-li ukázky spustit, buď použijte spustitelné verze dodané, nebo zkompilujte zdrojové verze jako jakékoli jiné aplikace produktu IBM MQ for Windows .

Postup

1. Vytvořte správce front a nastavte výchozí definice.

   Než budete moci spustit některý z ukázkových programů, musíte to provést. Další informace o vytváření správce front naleznete v tématu Administrace produktu IBM MQ. Informace o konfiguraci správce front tak, aby bezpečně přijímal požadavky na příchozí připojení od aplikací spuštěných v režimu klienta, viz “Konfigurace správce front pro příjem klientských připojení na platformách Multiplatforms” na stránce 1045.
2. Pokud nepoužíváte své vlastní fronty, spusťte ukázkový soubor MQSC amqscos0.tst pro vytvoření sady front.

   Chcete-li to provést v systémech Windows , zadejte:

   ```
   runmqsc QManagerName < amqscos0.tst > sampobj.out
   ```

   Zkontrolujte soubor `sampobj.out` , abyste se ujistili, že se nevyskytly žádné chyby. Tento soubor se nachází ve vašem aktuálním adresáři.

   Poznámka:
3. Chcete-li použít příklady jazyka COBOL pro příklady Inquire, Set a Echo, změňte definice procesu před spuštěním těchto ukázek.

   Pro příklady Inquire, Set a Echo spustí ukázkové definice verze C těchto ukázek. Chcete-li používat verze jazyka COBOL, musíte změnit definice procesu:

   - SYSTEM.SAMPLE.INQPROCESS
   - SYSTEM.SAMPLE.SETPROCESS
   - SYSTEM.SAMPLE.ECHOPROCESS

   V systému Windowsto provedete tak, že upravíte soubor `amqscos0.tst` a změníte názvy spustitelných souborů jazyka C na názvy spustitelných souborů v jazyce COBOL dříve, než použijete příkaz **runmqsc** ke spuštění těchto ukázek.
4. Spusťte ukázkové programy.

   Chcete-li spustit ukázku, zadejte její název následovaný libovolkovými parametry, například:

   ```
   amqsput myqueue qmanagername
   ```

   kde *myqueue* je název fronty, na kterou se mají zprávy vložit, a *qmanagername* je správce front, který vlastní *myqueue*.

   Další informace o parametrech, které každý z ukázek očekává, najdete v popisech jednotlivých ukázek.

   Poznámka: U ukázkových programů v jazyce COBOL při předávání názvů front jako parametrů je třeba zadat 48 znaků a v případě potřeby vyplňující prázdné znaky. Cokoli jiného než 48 znaků způsobí, že program selže s kódem příčiny 2085.

Související odkazy

“Ukázky pro IBM MQ for Windows” na stránce 1039
Techniky demonstrovány ukázkovými programy pro IBM MQ for Windows.

“Ukázky jazyka Visual Basic pro produkt IBM MQ for Windows” na stránce 1042
Techniky demonstrovány ukázkovými programy pro produkt IBM MQ v systémech Windows .

Ukázkový program uživatelské procedury rozhraní API

Ukázková uživatelská procedura rozhraní API vygeneruje trasování MQI do uživatelem určeného souboru s předponou definovanou v proměnné prostředí MQAPI_TRACE_LOGFILE.

Další informace o uživatelských procedurách rozhraní API najdete v tématu “Zápis a kompilace uživatelských procedur rozhraní API” na stránce 917.

Zdroj
amqsaxe0.c

Binární
amqsaxe

Konfigurace pro uživatelskou proceduru pro ukázku

1. Přidejte následující do souboru qm.ini .

   Platformy jiné než Windows

   ```
   ApiExitLocal:
   Sequence=100
   Function=EntryPoint
   Module= MQ_INSTALLATION_PATH/samp/bin/amqsaxe
   Name=SampleApiExit
   ```

   kde *MQ_INSTALLATION_PATH* představuje adresář, kde je nainstalován produkt IBM MQ .

   Windows

   ```
   ApiExitLocal:
   Sequence=100
   Function=EntryPoint
   Module= MQ_INSTALLATION_PATH\Tools\c\Samples\bin\amqsaxe
   Name=SampleApiExit
   ```

   kde *MQ_INSTALLATION_PATH* představuje adresář, kde je nainstalován produkt IBM MQ .

2. Nastavit proměnnou prostředí

   ```
   MQAPI_TRACE_LOGFILE=/tmp/MqiTrace
   ```

3. Spusťte aplikaci.

   Výstupní soubory jsou vytvářeny v adresáři /tmp s názvy jako: `MqiTrace.pid.tid.log`

Ukázkový program asynchronní spotřeby

Ukázkový program amqscbf demonstruje použití příkazů MQCB a MQCTL ke spotřebování zpráv z více front asynchronně.

amqscbf je k dispozici jako zdrojový kód jazyka C a binární klient a server na platformách Windows, UNIX and Linux .

Program je spuštěn z příkazového řádku a používá následující volitelné parametry:

```
Usage: [Options] Queue Name {queue_name}
  where Options are:
    -m Queue Manager Name
    -o Open options
    -r Reconnect Type
       d Reconnect Disabled
       r Reconnect
       m Reconnect Queue Manager
```

Chcete-li číst zprávy z více front, zadejte více než jeden název fronty (maximálně deset front je podporováno vzorkem).

Poznámka: Reconnect type je platné pouze pro klientské programy.

Příklad

Příklad zobrazuje příkaz amqscbf spuštěný jako serverový program, který čte jednu zprávu z produktu QL1 a poté je zastavován.

Pomocí Průzkumníku IBM MQ vložte testovací zprávu do produktu QL1. Zastavte program stisknutím klávesy Enter.

```
C:\>amqscbf QL1
Sample AMQSCBF0 start

Press enter to end
Message Call (9 Bytes) :
Message 1

Sample AMQSCBF0 end
```

Co parametr amqscbf demonstruje

Ukázka ukazuje, jak číst zprávy z více front v pořadí jejich příchodu. To bude vyžadovat mnohem více kódu pomocí synchronního příkazu MQGET. V případě asynchronní spotřeby není požadován žádný systém výzev a správa podprocesů a úložišť je prováděna produktem IBM MQ. Příklad "reálného světa" by se musel vypořádat s chybami; v ukázkových chybách jsou na konzolu odepsány chyby.

Vzorový kód má následující kroky:

1. Definujte jednotlivou funkci zpětného volání spotřeby zpráv,

   ```
   void MessageConsumer(MQHCONN      hConn,
                        MQMD    * pMsgDesc,
                        MQGMO   * pGetMsgOpts,
                        MQBYTE  * Buffer,
                        MQCBC   * pContext)
   { ... }
   ```

2. Připojte se ke správci front,

   ```
   MQCONNX(QMName,&cno,&Hcon,&CompCode,&CReason);
   ```

3. Otevřete vstupní fronty a přidružte je ke každé funkci zpětného volání `MessageConsumer` .

   ```
   MQOPEN(Hcon,&od,O_options,&Hobj,&OpenCode,&Reason);
   cbd.CallbackFunction = MessageConsumer;
   MQCB(Hcon,MQOP_REGISTER,&cbd,Hobj,&md,&gmo,&CompCode,&Reason);
   ```

   `cbd.CallbackFunction` není třeba nastavovat pro každou frontu; jedná se pouze o vstupní pole. Ke každé frontě byste však mohli přidružit jinou funkci zpětného volání.

4. Spustit spotřebu zpráv,

   ```
   MQCTL(Hcon,MQOP_START,&ctlo,&CompCode,&Reason);
   ```

5. Počkejte, až uživatel stiskne stisknutou klávesu Enter a poté ukončí spotřebu zpráv,

   ```
   MQCTL(Hcon,MQOP_STOP,&ctlo,&CompCode,&Reason);
   ```

6. Nakonec se odpojte od správce front,

   ```
   MQDISC(&Hcon,&CompCode,&Reason);
   ```

Ukázkový program Asynchronous Put

Zde se dozvíte o spuštění ukázky amqsapt a návrhu ukázkového programu Asynchronous Put.

Ukázkový program pro asynchronní vložení vkládá zprávy do fronty s použitím asynchronního volání MQPUT a poté načte informace o stavu pomocí volání MQSTAT. Název tohoto programu na různých platformách viz “Vlastnosti demonstrovány v ukázkových programech na platformách Multiplatforms” na stránce 1036 .

Spuštění ukázky amqsapt

Tento program může mít až 6 parametrů:

1. Název cílové fronty (povinné)
2. Název správce front (volitelný)
3. Volby otevření (volitelné)
4. Volby zavření (volitelné)
5. Název cílového správce front (volitelné)
6. Název dynamické fronty (volitelné)

Není-li určen správce front, připojí se parametr amqSapt k výchozímu správci front.

Návrh ukázkového programu Asynchronous Put

Program používá volání MQOPEN s dodanými volbami pro výstup, nebo s volbami MQOO_OUTPUT a MQOO_FAIL_IF_QUIESCING k otevření cílové fronty pro vkládání zpráv.

Pokud nemůže frontu otevřít, výstupem programu je chybová zpráva obsahující kód příčiny vrácený voláním MQOPEN. Chcete-li program ponechat jednoduchý, a to i při následných voláních MQI, program použije výchozí hodnoty pro celou řadu voleb.

Pro každý řádek vstupu program přečte text do vyrovnávací paměti a použije volání MQPUT s MQPMO_ASYNC_RESPONSE k vytvoření zprávy datagramu obsahující text této linky a asynchronně jej umístí do cílové fronty. Program pokračuje, dokud nedosáhne konce vstupu, nebo volání MQPUT selže. Pokud se program dostane na konec vstupu, zavře frontu pomocí volání MQCLOSE.

Program pak vydá volání MQSTAT, vrátí strukturu MQSTS a zobrazí zprávy obsahující počet úspěšně vložených zpráv, počet zpráv vložených s varováním a počet selhání.

Ukázkové programy Procházet

Při procházení ukázkových programů procházejte zprávy ve frontě pomocí volání MQGET.

Názvy těchto programů viz “Vlastnosti demonstrovány v ukázkových programech na platformách Multiplatforms” na stránce 1036 .

Návrh ukázkového programu pro procházení

Program otevře cílovou frontu pomocí volání MQOPEN s volbou MQOO_BROWSE. Pokud nemůže frontu otevřít, výstupem programu je chybová zpráva obsahující kód příčiny vrácený voláním MQOPEN.

Pro každou zprávu ve frontě používá program volání MQGET ke zkopírování zprávy z fronty a poté zobrazí data obsažená ve zprávě. Volání MQGET používá tyto volby:

PŘÍŠTĚ MQGMO_BROWSE_NEXT
Po volání MQOPEN je kurzor procházení umístěn logicky před první zprávou ve frontě, takže tato volba způsobí vrácení zprávy ***první*** při prvním vyvolání volání.

MQGMO_NO_WAIT
Program nečeká, pokud ve frontě nejsou žádné zprávy.

SOUBOR MQGMO_ACCEPT_TRUNCATED_MSG
Volání MQGET určuje vyrovnávací paměť pevné velikosti. Je-li zpráva delší než tato vyrovnávací paměť, program zobrazí oříznutou zprávu spolu s varováním, že zpráva byla zkrácena.

Tento program demonstruje, jak musíte vymazat pole *MsgId* a *CorrelId* struktury MQMD po každém volání MQGET, protože volání nastavuje tato pole na hodnoty obsažené ve zprávě, kterou načítá. Vymazání těchto polí znamená, že po sobě jdoucí příkazy MQGET načítají zprávy v pořadí, ve kterém jsou zprávy zadrženy ve frontě.

Program pokračuje na konci fronty; volání MQGET vrátí kód příčiny MQRC_NO_MSG_AVAILABLE a program zobrazí varovnou zprávu. Pokud se volání MQGET nezdaří, zobrazí program chybovou zprávu, která obsahuje kód příčiny.

Program poté zavře frontu pomocí volání MQCLOSE.

Vzorové programy Procházet pro UNIX, Linux, and Windows
Zvažte použití tohoto tématu, když se seznámíte s ukázkovými programy Procházet v produktu UNIX, Linux, and Windows.

Verze C programu má 2 parametry

1. Název zdrojové fronty (nezbytné)
2. Název správce front (volitelný)

Není-li správce front zadán, připojí se k výchozímu správci front. Zadejte například jednu z následujících možností:

- `amqsgbr myqueue qmanagername`
- `amqsgbrc myqueue qmanagername`
- `amq0gbr0 myqueue`

kde `myqueue` je název fronty, ze které se budou zobrazovat zprávy, a `qmanagername` je správce front, který vlastní `myqueue`.

Pokud při spuštění ukázky jazyka C vynecháte `qmanagername`, předpokládá se, že výchozí správce front vlastní tuto frontu.

Verze COBOL nemá žádné parametry. Připojuje se k výchozímu správci front, a když jej spustíte, budete vyzváni:

```
Please enter the name of the target queue
```

Zobrazí se pouze prvních 50 znaků každé zprávy, za nímž bude následovat `- - - truncated` , pokud se jedná o tento případ.

Ukázkové programy Procházet v systému IBM i
Každý program načte kopie všech zpráv ve frontě, které uvedete při volání programu; zprávy zůstanou ve frontě.

Je možné použít dodanou frontu SYSTEM.SAMPLE.LOCAL; nejprve spusťte vzorový program, abyste vložili nějaké zprávy do fronty. Můžete použít frontu SYSTEM.SAMPLE.ALIAS, což je název aliasu pro stejnou lokální frontu. Program pokračuje, dokud nedosáhne konce fronty, nebo se nezdaří volání MQI.

Ukázky jazyka C umožňují zadat název správce front, obvykle jako druhý parametr, obdobným způsobem jako ukázky systému produktu Windows . Příklad:

```
CALL PGM(QMQM/AMQSTRG4) PARM('SYSTEM.SAMPLE.TRIGGER' 'QM01')
```

Není-li správce front zadán, připojí se k výchozímu správci front. To je také důležité pro ukázky RPG. Při ukázkách jazyka RPG je však nutné zadat název správce front, a nikoli jeho výchozí nastavení.

ULW *Ukázkový program prohlížeče*

Ukázkový program prohlížeče čte a zapisuje jak deskriptor zprávy, tak pole obsahu zprávy ve všech zprávách ve frontě.

Ukázkový program je napsán jako utilita, ne jen aby demonstroval techniku. Názvy těchto programů viz "Vlastnosti demonstrovány v ukázkových programech na platformách Multiplatforms" na stránce 1036 .

Tento program vezme tyto poziční parametry:

1. Název zdrojové fronty (povinné)
2. Název správce front (povinné)
3. Volitelný parametr pro vlastnosti (volitelné).

Tyto programy také používají proměnnou prostředí s názvem **MQSAMP_USER_ID** , která by měla být nastavena na ID uživatele, které má být použito pro ověření připojení. Je-li nastavena tato volba, program vyzve k zadání hesla, které má být připojeno k tomuto ID uživatele.

Chcete-li spustit tyto programy, zadejte jeden z následujících příkazů:

- `amqsbcg` *`myqueue qmanagername`*
- `amqsbcgc` *`myqueue qmanagername`*

kde *myqueue* je název fronty, v níž mají být zprávy procházeny, a *qmanagername* je správce front, který vlastní *myqueue*.

Přečte každou zprávu z fronty a zapíše na standardní výstup následující výstup:

- Pole deskriptoru formátované zprávy
- Data zprávy (vypsána v hexadecimálním formátu a, je-li to možné, znakový formát)

Tabulka 151. Přípustné hodnoty pro parametr vlastnosti

Hodnota	Chování
0	Výchozí chování, jak bylo pro IBM WebSphere MQ 6. Vlastnosti, které se doručí do aplikace, závisí na atributu fronty produktu **PropertyControl** , ze kterého se zpráva načítá.

1	Popisovač zprávy je vytvořen a použit s parametrem MQGET. Vlastnosti zprávy s výjimkou těch, které jsou obsaženy v deskriptoru (či rozšíření) zprávy, jsou zobrazeny podobným způsobem jako deskriptor zprávy. Příklad:
	`****Message properties****`
	`  property name: property value`
	Nebo nejsou-li k dispozici žádné vlastnosti:
	`****Message properties****`
	`  None`
	Číselné hodnoty jsou zobrazeny pomocí `printf`, hodnoty řetězce jsou uzavřeny v jednoduchých uvozovkách a bajtové řetězce jsou obklopeny X a jednoduchými uvozovkami, jako pro deskriptor zprávy.

2	Hodnota MQGMO_NO_PROPERTIES je určena tak, aby byly vráceny pouze vlastnosti deskriptoru zpráv.
3	Je zadán parametr MQGMO_PROPERTIES_FORCE_MQRFH2 , takže všechny vlastnosti jsou vráceny v datech zprávy.
4	Funkce MQGMO_PROPERTIES_COMPATIBILITY je určena tak, aby všechny vlastnosti mohly být vráceny v závislosti na tom, zda je vlastnost IBM WebSphere MQ 6 zahrnuta, jinak jsou vlastnosti zrušeny.

Program je omezen na tisk prvních 65535 znaků zprávy a selže s příčinou `truncated msg` , pokud je přečtena delší zpráva.

Příklad výstupu tohoto obslužného programu naleznete v tématu .

Ukázka transakce CICS

Ukázkový transakční program CICS je poskytován s názvem amqscic0.ccs pro zdrojový kód a amqscic0 pro spustitelnou verzi. Transakce můžete sestavovat pomocí standardních zařízení produktu CICS .

Podrobnosti o příkazech potřebných pro vaši platformu naleznete v příručce “Sestavení procedurální aplikace” na stránce 966 .

Transakce čte zprávy z přenosové fronty SYSTEM.SAMPLE.CICS.WORKQUEUE na výchozím správci front a umisťuje je do lokální fronty, jehož název je obsažen v záhlaví přenosu zprávy. Veškerá selhání se odesílají do fronty SYSTEM.SAMPLE.CICS.DLQ.

Poznámka: K vytvoření těchto front a ukázkových vstupních front můžete použít vzorový skript MQSC amqscic0.tst .

Ukázkový program Connect

Ukázkový program Connect vám umožňuje prozkoumat volání MQCONNX a jeho volby od klienta. Ukázka se připojí ke správci front pomocí volání MQCONNX, inquires o názvu správce front s použitím volání MQINQ a zobrazí jej. Také se seznámíte se spuštěním ukázky amqscnxc.

Poznámka: Ukázkový program Connect je ukázkový klient. Můžete ji zkompilovat a spustit na serveru, ale funkce je smysluplná pouze na straně klienta a jsou dodány pouze spustitelné soubory klienta.

Spuštění ukázky amqscnxc

Syntaxe příkazového řádku ukázkového programu Connect je následující:

```
amqscnxc [-x ConnName [-c SvrconnChannelName]] [-u User] [QMgrName]
```

Parametry jsou volitelné a jejich pořadí není důležité, kromě parametru QMgrName, který, je-li zadán, musí být poslední. Parametry jsou:

ConnName
Název připojení TCP/IP správce front serveru

Nezadáte-li název připojení TCP/IP, bude MQCONNX zadán spolu s parametrem *ClientConnPtr* nastaveným na hodnotu NULL.

Název SvrconnChannel
Název kanálu připojení serveru

Uvedete-li jméno připojení TCP/IP, ale ne kanál připojení k serveru (zpětné nastavení není povoleno), ukázka použije název SYSTEM.DEF.SVRCONN.

Uživatel
Jméno uživatele, které má být použito pro ověření připojení

Uvedete-li tento program, zobrazí se výzva k zadání hesla, které má být připojeno k tomuto ID uživatele.

QMgrName
Název cílového správce front

Pokud neurčíte cílového správce front, bude se ukázka připojovat k kterémkoliv správci front, který naslouchá danému názvu připojení protokolu TCP/IP.

Poznámka: Zadáte-li otazník jako jediný parametr nebo zadáte-li nesprávné parametry, zobrazí se zpráva s vysvětlením, jak tento program používat.

Spustíte-li ukázku bez voleb příkazového řádku, bude obsah proměnné prostředí MQSERVER použit k určení informací o připojení. (V tomto příkladu je MQSERVER nastaven na `SYSTEM.DEF.SVRCONN/TCP/machine.site.company.com`.) Zobrazí se výstup podobný tomuto:

```
Sample AMQSCNXC start
Connecting to the default queue manager
with no client connection information specified.
Connection established to queue manager machine

Sample AMQSCNXC end
```

Pokud spustíte ukázku a zadáte-li název připojení TCP/IP a název kanálu připojení serveru, ale žádný název cílového správce front, jako je tento:

```
amqscnxc -x machine.site.company.com -c SYSTEM.ADMIN.SVRCONN
```

je použit výchozí název správce front a vidíte výstup podobný tomuto:

```
Sample AMQSCNXC start
Connecting to the default queue manager
using the server connection channel SYSTEM.ADMIN.SVRCONN
on connection name machine.site.company.com.
Connection established to queue manager MACHINE

Sample AMQSCNXC end
```

Pokud spustíte ukázku a zadáte-li název připojení TCP/IP a název cílového správce front, jako je tento:

```
amqscnxc -x machine.site.company.com MACHINE
```

zobrazí se výstup podobný tomuto:

```
Sample AMQSCNXC start
Connecting to queue manager MACHINE
using the server connection channel SYSTEM.DEF.SVRCONN
on connection name machine.site.company.com.
Connection established to queue manager MACHINE

Sample AMQSCNXC end
```

Ukázkový program pro převod dat

Ukázkový program pro převod dat je kostra uživatelské rutiny pro převod dat. Informace o návrhu ukázky pro převod dat.

Názvy těchto programů viz “Vlastnosti demonstrovány v ukázkových programech na platformách Multiplatforms” na stránce 1036 .

Návrh souboru pro převod dat

Každá uživatelská procedura pro převod dat převádí jeden pojmenovaný formát zpráv. Tato kostra je určena jako obálka pro fragmenty kódu generované obslužným programem pro generování dat ukončení převodu dat.

Obslužný program vytváří jeden fragment kódu pro každou strukturu dat; několik takových struktur tvoří formát, takže se do této kostry přidá několik fragmentů kódu, aby bylo možné vytvořit rutinu tak, aby bylo možné provádět převod dat celého formátu.

Program potom zkontroluje, zda je konverze úspěšná nebo neúspěšná, a vrátí hodnoty vyžadované volajícímu.

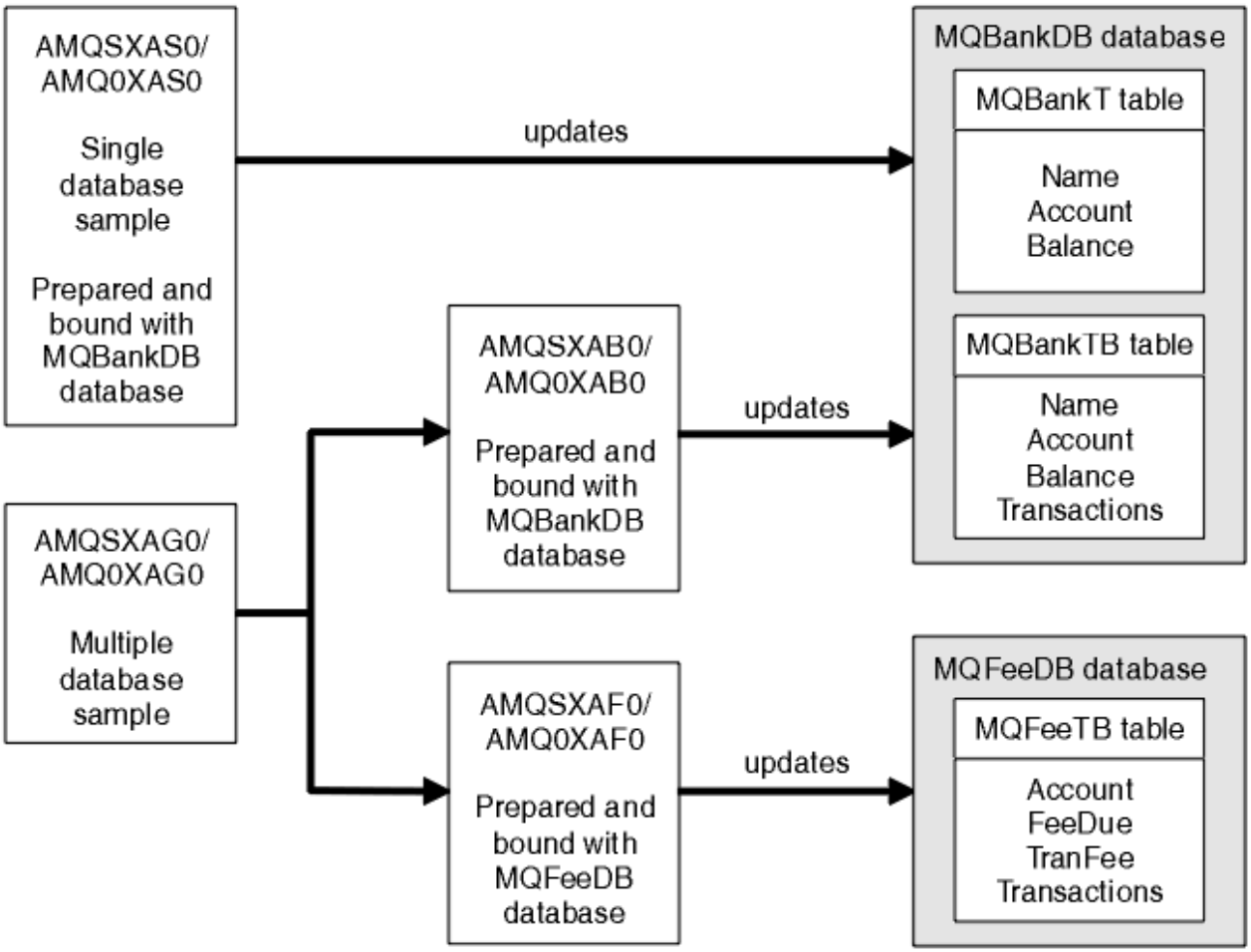
Ukázky koordinace databází

K dispozici jsou dva ukázky, které ukazují, jak může produkt IBM MQ koordinovat jak aktualizace produktu IBM MQ , tak aktualizace databáze v rámci stejné jednotky práce.

Tyto vzorky jsou:

1. AMQSXAS0 (v jazyce C) nebo AMQ0XAS0 (v COBOLu), které aktualizuje jednu databázi v rámci pracovní jednotky IBM MQ .
2. AMQSXAG0 (v jazyce C) nebo AMQ0XAG0 (v COBOLu), AMQSXAB0 (v jazyce C) nebo AMQ0XAB0 (v COBOLu), a AMQSXAF0 (v jazyce C) nebo AMQ0XAF0 (v COBOLu), které dohromady aktualizují dvě databáze v rámci pracovní jednotky IBM MQ , které ukazují, jak lze přistupovat k více databázím. Tyto ukázky jsou k dispozici pro zobrazení použití volání MQBEGIN, smíšených volání SQL a IBM MQ a kde a kdy a kdy se má připojit k databázi.

Produkt Obrázek 140 na stránce 1058 zobrazuje, jak jsou použité ukázky použity k aktualizaci databází:



Obrázek 140. Ukázky koordinace databází

Programy přečtou zprávu z fronty (pod synchronizačním bodem) a pak pomocí informací ve zprávě získají příslušné informace z databáze a aktualizují ji. Pak se vytiskne nový stav databáze.

Logika programu je následující:

1. Použijte název vstupní fronty z argumentu programu
2. Připojit se k výchozímu správci front (nebo volitelně k zadanému názvu v jazyce C) pomocí MQCONN
3. Otevření fronty (použití funkce MQOPEN) pro vstup při neúspěších
4. Spuštění jednotky práce pomocí MQBEGIN
5. Získat další zprávu (pomocí příkazu MQGET) z fronty pod synchronizačním bodem

6. Získat informace z databází
7. Aktualizovat informace z databází
8. Potvrdit změny pomocí MQCMIT
9. Vytisknout aktualizované informace (žádná zpráva, která má být k dispozici, se počítá jako selhání, a konec smyčky)
10. Zavřít frontu pomocí příkazu MQCLOSE
11. Odpojit od fronty pomocí MQDISC

Kurzory SQL se používají ve vzorcích, takže čtení z databází (tj. více instancí) je uzamčeno během zpracování zprávy, což umožňuje simultánní spuštění více instancí těchto programů. Kurzory jsou explicitně otevřeny, ale volání MQCMIT je implicitně zavřeny.

Ukázka jedné databáze (AMQSXAS0 nebo AMQ0XAS0) nemá žádné příkazy SQL CONNECT a připojení k databázi je implicitně vytvořeno produktem IBM MQ s voláním MQBEGIN. Ukázka více databází (AMQSXAG0 nebo AMQ0XAG0, AMQSXAB0 nebo AMQ0XAB0a AMQSXAF0 nebo AMQ0XAF0) obsahuje příkazy SQL CONNECT, protože některé databázové produkty povolují pouze jedno aktivní připojení. Pokud se nejedná o tento případ pro daný databázový produkt nebo pokud přistupujete k jedné databázi ve více databázových produktech, lze příkazy SQL CONNECT odstranit.

Ukázky jsou připravovány s databázovým produktem IBM Db2 , takže je možná budete muset upravit tak, aby fungovaly s dalšími databázovými produkty.

Kontrola chyb SQL používá rutiny v UTIL.C a CHECKERR.CBL dodávala Db2. Musí být kompilovány nebo nahrazeny před kompilací a propojením.

Poznámka: Pokud používáte zdroj jazyka COBOL Micro Focus, CHECKERR.MFC pro kontrolu chyb SQL, musíte změnit ID programu na velká písmena, což je CHECKERR, aby AMQ0XAS0 propojí správně.

Vytvoření databází a tabulek
Vytvořte databáze a tabulky před kompilací ukázek.

Chcete-li vytvořit databáze, použijte obvyklou metodu pro váš databázový produkt, například:

```
DB2 CREATE DB MQBankDB
DB2 CREATE DB MQFeeDB
```

Vytvořte tabulky pomocí příkazů SQL následujícím způsobem:

V C:

```
EXEC SQL CREATE TABLE MQBankT(Name         VARCHAR(40) NOT NULL,
                              Account      INTEGER     NOT NULL,
                              Balance      INTEGER     NOT NULL,
                              PRIMARY KEY (Account));

EXEC SQL CREATE TABLE MQBankTB(Name         VARCHAR(40) NOT NULL,
                               Account      INTEGER     NOT NULL,
                               Balance      INTEGER     NOT NULL,
                               Transactions INTEGER,
                               PRIMARY KEY (Account));

EXEC SQL CREATE TABLE MQFeeTB(Account      INTEGER     NOT NULL,
                              FeeDue       INTEGER     NOT NULL,
                              TranFee      INTEGER     NOT NULL,
                              Transactions INTEGER,
                              PRIMARY KEY (Account));
```

V jazyce COBOL:

```
EXEC SQL CREATE TABLE
  MQBankT(Name         VARCHAR(40) NOT NULL,
          Account      INTEGER     NOT NULL,
          Balance      INTEGER     NOT NULL,
          PRIMARY KEY (Account))
  END-EXEC.
```

```
EXEC SQL CREATE TABLE
  MQBankTB(Name         VARCHAR(40) NOT NULL,
           Account      INTEGER     NOT NULL,
           Balance      INTEGER     NOT NULL,
           Transactions INTEGER,
           PRIMARY KEY (Account))
  END-EXEC.

EXEC SQL CREATE TABLE
  MQFeeTB(Account      INTEGER   NOT NULL,
          FeeDue       INTEGER   NOT NULL,
          TranFee      INTEGER   NOT NULL,
          Transactions INTEGER,
          PRIMARY KEY (Account))
  END-EXEC.
```

Zadejte data do tabulek pomocí příkazů SQL následujícím způsobem:

```
EXEC SQL INSERT INTO MQBankT VALUES ('Mr Fred Bloggs',1,0);
EXEC SQL INSERT INTO MQBankT VALUES ('Mrs S Smith',2,0);
EXEC SQL INSERT INTO MQBankT VALUES ('Ms Mary Brown',3,0);
⋮
EXEC SQL INSERT INTO MQBankTB VALUES ('Mr Fred Bloggs',1,0,0);
EXEC SQL INSERT INTO MQBankTB VALUES ('Mrs S Smith',2,0,0);
EXEC SQL INSERT INTO MQBankTB VALUES ('Ms Mary Brown',3,0,0);
⋮
EXEC SQL INSERT INTO MQFeeTB VALUES (1,0,50,0);
EXEC SQL INSERT INTO MQFeeTB VALUES (2,0,50,0);
EXEC SQL INSERT INTO MQFeeTB VALUES (3,0,50,0);
⋮
```

Poznámka: Pro COBOL použijte stejné příkazy SQL, ale přidejte END_EXEC na konec každého řádku.

Předkompilace, kompilace a propojování ukázek
Seznamte se s předkompilací, kompilací a propojením ukázek v jazycích C a COBOL.

Předkompilujte soubory .SQC (v jazyce C) a soubory .SQB (v jazyce COBOL) a svážete je s příslušnou databází pro vytvoření souborů .C nebo .CBL. Chcete-li to provést, použijte typickou metodu pro váš databázový produkt.

Předkompilace v jazyce C

```
db2 connect to MQBankDB
db2 prep AMQSXAS0.SQC
db2 connect reset

db2 connect to MQBankDB
db2 prep AMQSXAB0.SQC
db2 connect reset

db2 connect to MQFeeDB
db2 prep AMQSXAF0.SQC
db2 connect reset
```

Předkompilace v jazyce COBOL

```
db2 connect to MQBankDB
db2 prep AMQ0XAS0.SQB bindfile target ibmcob
db2 bind AMQ0XAS0.BND
db2 connect reset

db2 connect to MQBankDB
db2 prep AMQ0XAB0.SQB bindfile target ibmcob
db2 bind AMQ0XAB0.BND
db2 connect reset

db2 connect to MQFeeDB
db2 prep AMQ0XAF0.SQB bindfile target ibmcob
```

```
db2 bind AMQ0XAF0.BND
db2 connect reset
```

Kompilace a propojení

V následujících ukázkových příkazech jsou použity symboly *DB2TOP* a *MQ_INSTALLATION_PATH*. *DB2TOP* představuje instalační adresář pro produkt Db2 . *MQ_INSTALLATION_PATH* představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

- V systému AIXje cesta k adresáři:

```
/usr/lpp/db2_05_00
```

- V systémech HP-UX a Solarisje cesta k adresáři následující:

```
/opt/IBMdb2/V5.0
```

- Na systémech Windows závisí cesta k adresáři na cestě zvolené při instalaci produktu. Pokud jste zvolili výchozí nastavení, je cesta následující:

```
c:\sqllib
```

Poznámka: Před zadáním příkazu link na systémech Windows se ujistěte, že proměnná prostředí LIB obsahuje cesty k knihovnám Db2 a IBM MQ .

Zkopírujte následující soubory do dočasného adresáře:

- Soubor `amqsxag0.c` z instalace produktu IBM MQ .

  Poznámka: Tento soubor lze nalézt v následujících adresářích:

  – V systémech UNIX and Linux:

  ```
  MQ_INSTALLATION_PATH/samp/xatm
  ```

  – V systémech Windows:

  ```
  MQ_INSTALLATION_PATH\tools\c\samples\xatm
  ```

- Soubory `.c` , které jste získali předkompilací zdrojových souborů `.sqc` , `amqsxas0.sqc` , `amqsxaf0.sqc`a `amqsxab0.sqc`
- Soubory `util.c` a `util.h` z vaší instalace Db2 .

  Poznámka: Tyto soubory lze nalézt v adresáři:

  ```
  DB2TOP/samples/c
  ```

Sestavte objektové soubory pro každý soubor `.c` pomocí následujícího příkazu kompilátoru pro platformu, kterou používáte:

- AIX

```
xlc_r -I MQ_INSTALLATION_PATH/inc -I DB2TOP/include -c -o
FILENAME.o FILENAME.c
```

- HP-UX

```
cc -Aa +z -I MQ_INSTALLATION_PATH/inc -I DB2TOP/include -c -o
FILENAME.o FILENAME.c
```

- Solaris

```
cc -Aa -KPIC -mt -I MQ_INSTALLATION_PATH
/inc -I DB2TOP/include -c -o
FILENAME.o FILENAME.c
```

- Systémy Windows

```
cl /c /I MQ_INSTALLATION_PATH\tools\c\include /I DB2TOP\include
FILENAME.c
```

Sestavte spustitelný soubor produktu `amqsxag0` pomocí následujícího příkazu odkazu pro platformu, kterou používáte:

- AIX

```
xlc_r -H512 -T512 -L DB2TOP/lib -ldb2 -L MQ_INSTALLATION_PATH/lib
-lmqm util.o amqsxaf0.o amqsxab0.o amqsxag0.o -o amqsxag0
```

- HP-UX Revize 11i

```
ld -E -L DB2TOP/lib -ldb2 -L MQ_INSTALLATION_PATH/lib -lmqm -lc -lpthread -lcl
/lib/crt0.o util.o amqsxaf0.o amqsxab0.o amqsxag0.o -o amqsxag0
```

- Solaris

```
cc -mt -L DB2TOP/lib -ldb2 -L MQ_INSTALLATION_PATH/lib
-lmqm -lthread -lsocket -lc -lnsl -ldl util.o
amqsxaf0.o amqsxab0.o amqsxag0.o -o amqsxag0
```

- Systémy Windows

```
link util.obj amqsxaf0.obj amqsxab0.obj amqsxag0.obj mqm.lib db2api.lib
/out:amqsxag0.exe
```

Sestavte spustitelný soubor produktu `amqsxas0` s použitím následujících příkazů pro kompilaci a propojení pro platformu, kterou používáte:

- AIX

```
xlc_r -H512 -T512 -L DB2TOP/lib -ldb2
-L MQ_INSTALLATION_PATH/lib -lmqm util.o amqsxas0.o -o amqsxas0
```

- HP-UX Revize 11i

```
ld -E -L DB2TOP/lib -ldb2 -L MQ_INSTALLATION_PATH/lib -lmqm -lc -lpthread
-lcl /lib/crt0.o util.o amqsxas0.o -o amqsxas0
```

- Solaris

```
cc -mt -L DB2TOP/lib -ldb2-L MQ_INSTALLATION_PATH/lib
-lqm -lthread -lsocket -lc -lnsl -ldl util.o
amqsxas0.o -o amqsxas0
```

- Systémy Windows

```
link util.obj amqsxas0.obj mqm.lib db2api.lib /out:amqsxas0.exe
```

Další informace

Pokud pracujete s produktem AIX nebo HP-UX a chcete přistupovat k databázi Oracle, použijte kompilátor xlc_r a odkaz na soubor libmqm_r.a.

Spuštění ukázek
V této části se dozvíte, jak nakonfigurovat správce front před spuštěním ukázek koordinace databází v jazycích C a COBOL.

Před spuštěním ukázek nakonfigurujte správce front s použitím databázového produktu, který používáte. Další informace o tom, jak to provést, najdete v tématu Scénář 1: Správce front provádí koordinaci.

Následující titulky poskytují informace o tom, jak spouštět ukázky v jazycích C a COBOL:

- "Ukázky jazyka C" na stránce 1063
- "Ukázky jazyka COBOL" na stránce 1064

Ukázky jazyka C

Zprávy musí být v následujícím formátu, aby mohly být čteny z fronty:

```
UPDATE Balance change=nnn WHERE Account=nnn
```

AMQSPUT lze použít k umístění zpráv do fronty.

Ukázky koordinace databází mají dva parametry:

1. Název fronty (povinné)
2. Název správce front (volitelný)

Za předpokladu, že jste vytvořili a nakonfigurovali správce front pro jednu ukázku databáze s názvem singDBQM, s frontou s názvem singDBQ, zvyšte hodnotu účtu pana Fred Bloggs o 50 takto:

```
AMQSPUT singDBQ singDBQM
```

Pak klíč v této zprávě:

```
UPDATE Balance change=50 WHERE Account=1
```

Do fronty můžete vložit více zpráv.

```
AMQSXAS0 singDBQ singDBQM
```

Pak se vytiskne aktualizovaný stav účtu pana Freda Bloggse.

Za předpokladu, že jste vytvořili a nakonfigurovali správce front pro ukázku více databází s názvem multDBQM, s frontou nazvanou multDBQ, snižujete účet paní Mary Brownové o 75 takto:

```
AMQSPUT multDBQ multDBQM
```

Pak klíč v této zprávě:

```
UPDATE Balance change=-75 WHERE Account=3
```

Do fronty můžete vložit více zpráv.

```
AMQSXAG0 multDBQ multDBQM
```

Pak se vytiskne aktualizovaný stav účtu paní Mary Brownové.

Ukázky jazyka COBOL

Zprávy musí být v následujícím formátu, aby mohly být čteny z fronty:

```
UPDATE Balance change=snnnnnnnn WHERE Account=nnnnnnnn
```

Pro jednoduchost musí `Balance change` být číslo se znakem osmdesát a `Account` musí být osminásobné číslo.

Ukázku AMQSPUT lze použít k umístění zpráv do fronty.

Ukázky nepřijímají žádné parametry a používají výchozího správce front. Lze jej nakonfigurovat tak, aby v libovolném okamžiku mohl být spuštěn pouze jeden z ukázek. Za předpokladu, že jste nakonfigurovali výchozího správce front pro jednu ukázku databáze s frontou s názvem singDBQ, zvyšte hodnotu účtu pana Fred Bloggs o 50 takto:

```
AMQSPUT singDBQ
```

Pak klíč v této zprávě:

```
UPDATE Balance change=+00000050 WHERE Account=00000001
```

Do fronty můžete vložit více zpráv:

```
AMQ0XAS0
```

Zadejte název fronty:

```
singDBQ
```

Pak se vytiskne aktualizovaný stav účtu pana Freda Bloggse.

Za předpokladu, že jste nakonfigurovali výchozího správce front pro více databázových ukázek s frontou s názvem multDBQ, jste odhadli účet paní Mary Mary o 75 takto:

```
AMQSPUT multDBQ
```

Pak klíč v této zprávě:

```
UPDATE Balance change=-00000075 WHERE Account=00000003
```

Do fronty můžete vložit více zpráv:

```
AMQ0XAG0
```

Zadejte název fronty:

```
multDBQ
```

Pak se vytiskne aktualizovaný stav účtu paní Mary Brownové.

Ukázka obslužné rutiny fronty nedoručených zpráv

Je k dispozici ukázkový popisovač fronty nedoručených zpráv, název spustitelné verze je amqsdlq. Chcete-li používat obslužnou rutinu fronty nedoručených zpráv, která se liší od proměnné RUNMQDLQ, je zdroj ukázky k dispozici pro použití jako základ.

Ukázka je podobná obslužnému programu na dead-letter v produktu, ale trasování a hlášení chyb se liší. K dispozici jsou dvě proměnné prostředí:

TRASOVÁNÍ DOQ_TRACE
Nastavit na YES nebo ano, chcete-li zapnout trasování

ODQ_ZPR
Nastavte na název souboru obsahujícího chybové a informační zprávy. Poskytnutý soubor se nazývá amqsdlq.msg.

Tyto proměnné je třeba v závislosti na platformě zobrazit pomocí příkazů **export** nebo **set** , a to v závislosti na platformě; trasování je vypnuto pomocí příkazu **unset** .

Soubor s chybovou zprávou amqsdlq.msgmůžete upravit tak, aby vyhovoval vašim požadavkům. Ukázka vkládá zprávy do souboru stdout, **not** do souboru protokolu chyb IBM MQ .

Příručka Správa nebo *System Management Guide* pro vaši platformu vysvětluje, jak funguje obslužná rutina dead-letter a jak ji spouštíte.

Ukázkový program Distribuční seznam

Ukázka distribučního seznamu amqsptl0 poskytuje příklad vložení zprávy do několika front zpráv. Je založena na vzorku MQPUT amqsput0.

Spuštění ukázky distribučního seznamu, amqsptl0

Ukázka Distribuční seznam se spustí podobným způsobem jako ukázky Put.

Je třeba provést následující parametry:

- Názvy front
- Názvy správců front

Tyto hodnoty jsou zadány jako dvojice. Příklad:

```
amqsptl0 queue1 qmanagername1 queue2 qmanagername2
```

Fronty jsou otevírány pomocí MQOPEN a zprávy jsou vloženy do front pomocí příkazu MQPUT. Kódy příčiny jsou vraceny, pokud nejsou rozpoznány některé názvy front nebo správců front.

Nezapomeňte definovat kanály mezi správci front, aby mezi nimi mohly být zprávy toku zpráv. Ukázkový program to pro vás neprovede.

Návrh ukázky distribučního seznamu

umístění záznamů zpráv (MQPMRs) uvádí atributy zpráv pro každé místo určení. Ukázka poskytuje hodnoty pro *MsgId* a *CorrelId*a tyto hodnoty přepíší hodnoty uvedené ve struktuře MQMD.

Pole *PutMsgRecFields* ve struktuře MQPMO označuje, která pole se nacházejí v MQPMRs:

```
MQLONG PutMsgRecFields=MQPMRF_MSG_ID + MQPMRF_CORREL_ID;
```

Dále ukázka přidělí záznamy odezvy a záznamy objektů. Záznamy objektů (MQORs) vyžadují alespoň jeden pár názvů a sudý počet názvů, to jest, *ObjectName* a *ObjectQMgrName*.

Další fáze zahrnuje připojení ke správcům front pomocí volání MQCONN. Ukázka se pokusí o připojení ke správci front přidruženému k první frontě v MQOR; pokud toto selže, projde záznamy objektů na oplátku. Jste informováni o tom, že není možné připojit se k žádnému správci front a program se ukončí.

Cílové fronty se otevřou pomocí MQOPEN a zpráva je vložena do těchto front pomocí příkazu MQPUT. Všechny problémy a selhání jsou hlášeny v záznamech odpovědí (MQRRs).

Nakonec jsou cílové fronty uzavřeny pomocí funkce MQCLOSE a program se odpojí od správce front pomocí funkce MQDISC. Stejné záznamy odpovědí se používají pro každé volání, které uvádí *CompCode* a *Reason*.

Ukázkové programy Echo

Ukázkové programy Echo echo zprávy z fronty zpráv do fronty odpovědí.

Názvy těchto programů viz “Vlastnosti demonstrovány v ukázkových programech na platformách Multiplatforms” na stránce 1036 .

Programy jsou určeny ke spouštění jako spouštěné programy.

Na systémech IBM i, UNIX, Linux, and Windows je jejich jediným vstupem struktura MQTMC2 (zpráva spouštěče), která obsahuje název cílové fronty a správce front. Verze COBOL používá výchozího správce front.

IBM i V systému IBM ise ujistěte, že ukázkový program Echo, který chcete použít, je spuštěn zprávami přicházejícími do fronty SYSTEM.SAMPLE.ECHO. Chcete-li tak učinit, zadejte název ukázkového programu Echo, který chcete použít, v poli *ApplId* definice procesu SYSTEM.SAMPLE.ECHOPROCESS. (Pro tento příklad můžete použít příkaz CHGMQMPRC; podrobnosti viz Změna procesu MQ (CHGMQMPRC).) Ukázková fronta má typ spouštěče FIRST, takže pokud již ve frontě existují zprávy, než spustíte ukázku požadavku, ukázka Echo se nespustí pomocí zpráv, které odešlete.

Pokud jste definici správně nastavili, nejprve spusťte AMQSERV4 v jedné úloze a poté spusťte příkaz AMQSREQ4 v jiném. Můžete použít AMQSTRG4 místo AMQSERV4, ale potenciální zpoždění při odeslání úlohy by mohla méně snadno sledovat, co se děje.

Ukázkové programy požadavku slouží k odesílání zpráv do fronty SYSTEM.SAMPLE.ECHO. Ukázkové programy Echo odešlou zprávu s odpovědí obsahující data ve zprávě požadavku do fronty odpovědi určené ve zprávě s požadavkem.

Návrh ukázkových programů Echo

Program otevře frontu uvedenou ve struktuře zpráv spouštěče, která byla předána, když byla spuštěna. (Pro srozumitelnost se na tuto žádost odkazuje jako na frontu požadavků.) Program používá volání MQOPEN k otevření této fronty pro sdílený vstup.

Program používá volání MQGET k odstranění zpráv z této fronty. Toto volání používá volby MQGMO_ACCEPT_TRUNCATED_MSG, MQGMO_CONVERT a MQGMO_WAIT, s intervalem čekání 5 sekund. Program testuje deskriptor každé zprávy za účelem zjištění, zda se jedná o zprávu požadavku; pokud není, program zahodí zprávu a zobrazí varovnou zprávu.

Pro každý řádek vstupu program pak přečte text do vyrovnávací paměti a pomocí volání MQPUT1 vloží do fronty pro odpověď zprávu obsahující text této řádky.

Pokud se volání MQGET nezdaří, program vloží zprávu s hlášením do fronty odpovědí, přičemž nastaví pole *Feedback* deskriptoru zpráv na kód příčiny vrácený příkazem MQGET.

Pokud ve frontě požadavků nejsou žádné zprávy, program tuto frontu zavře a odpojí se od správce front.

IBM i V systému IBM imůže program také odpovídat na zprávy odeslané do fronty z jiných platforem než IBM MQ for IBM i, ačkoli pro tuto situaci není k dispozici žádný vzorek. Chcete-li provést práci programu ECHO:

- Napište program, který správně uvádí parametry **Format**, **Encoding**a **CCSID** , abyste odeslali zprávy s požadavky textu.

  Program ECHO požaduje, aby správce front provedl převod dat zpráv, pokud je to potřeba.
- Uveďte CONVERT (*YES) na odesílajícím kanálu IBM MQ for IBM i , pokud program, který jste napsali, nezajišťuje podobnou konverzi pro odpověď.

Ukázkové programy Get

Vzorové programy typu Get získají zprávy z fronty pomocí volání MQGET.

Názvy těchto programů viz “Vlastnosti demonstrovány v ukázkových programech na platformách Multiplatforms” na stránce 1036 .

Návrh ukázkového programu Get

Program otevře cílovou frontu pomocí volání MQOPEN s volbou MQOO_INPUT_AS_Q_DEF. Nemůže-li frontu otevřít, zobrazí se v programu chybová zpráva obsahující kód příčiny vrácený voláním MQOPEN.

Pro každou zprávu ve frontě program používá volání MQGET k odebrání zprávy z fronty a poté zobrazí data obsažená ve zprávě. Volání MQGET používá volbu MQGMO_WAIT a určuje *WaitInterval* o 15 sekund, takže program čeká na toto období, pokud ve frontě není žádná zpráva. Pokud před uplynutím tohoto intervalu nepřijde žádná zpráva, volání se nezdaří a vrátí kód příčiny MQRC_NO_MSG_AVAILABLE.

Tento program demonstruje, jak musíte vymazat pole *MsgId* a *CorrelId* struktury MQMD po každém volání MQGET, protože volání nastavuje tato pole na hodnoty obsažené ve zprávě, kterou načítá. Vymazání těchto polí znamená, že po sobě jdoucí příkazy MQGET načítají zprávy v pořadí, ve kterém jsou zprávy zadrženy ve frontě.

Volání MQGET určuje vyrovnávací paměť pevné velikosti. Je-li zpráva delší než tato vyrovnávací paměť, volání selže a program se zastaví.

Program pokračuje, dokud buď volání MQGET nevrátí kód příčiny MQRC_NO_MSG_AVAILABLE, nebo se volání MQGET nezdaří. Pokud se volání nezdaří, zobrazí program chybovou zprávu, která obsahuje kód příčiny.

Program poté zavře frontu pomocí volání MQCLOSE.

Spuštění ukázek amqsget a amqsgetc

Tyto programy používají následující poziční parametry:

1. Název zdrojové fronty (povinné)
2. Název správce front (volitelný)

   Není-li správce front zadán, funkce amqsget se připojí k výchozímu správci front a funkce amqsgetc se připojí ke správci front určenému proměnnou prostředí nebo definičním souborem kanálu klienta.
3. Volby otevření (volitelné)

   Nejsou-li volby otevření zadány, použije vzorek hodnotu 8193, což je kombinace těchto dvou voleb:

   - MQO_INPUT_AS_Q_DEF
   - UVÁDĚNÍ MQOO_FAIL_IF_QUIESCING
4. Volby zavření (volitelné)

   Nejsou-li volby uzavření zadány, bude v ukázce použita hodnota 0, což je MQCO_NONE.

Tyto programy také používají proměnnou prostředí s názvem **MQSAMP_USER_ID** , která by měla být nastavena na ID uživatele, které má být použito pro ověření připojení. Je-li nastavena tato volba, program zobrazí výzvu k zadání hesla, které má být připojeno k tomuto ID uživatele.

Chcete-li spustit tyto programy, zadejte jednu z následujících možností:

- `amqsget myqueue qmanagername`
- `amqsgetc myqueue qmanagername`

kde `myqueue` je název fronty, ze které bude program získávat zprávy, a `qmanagername` je správce front, který vlastní `myqueue`.

Použití amqsget a amqsgetc

Povšimněte si, že produkt **amqsget** provádí lokální připojení ke správci front s použitím sdílené paměti pro připojení ke správci front a jako takové lze spustit pouze v systému, ve kterém je správce front umístěn, zatímco produkt **amqsgetc** provádí připojení ve stylu klienta (i v případě připojení ke správci front ve stejném systému).

Při použití produktu **amqsgetc** je třeba poskytnout podrobné informace o aplikaci, jak skutečně dosáhnout správce front, pokud jde o hostitele správce front nebo adresu IP a port modulu listener správce front.

Obvykle je to provedeno buď pomocí proměnné prostředí MQSERVER, nebo definováním podrobností připojení pomocí tabulky definic kanálů klienta, kterou lze poskytnout také produktu **amqsgetc** pomocí proměnných prostředí; například viz MQCCDTURL.

Příklad použití MQSERVER, připojení ke správci front lokálně, který má modul listener spuštěný na portu 1414 a použití výchozího kanálu připojení serveru, je následující:

```
export MQSERVER="SYSTEM.DEF.SVRCONN/TCP/ localhost(1414)"
```

Ukázkové programy s vysokou dostupností

Ukázkové programy s vysokou dostupností **amqsghac**, **amqsphac**a **amqsmhac** používají automatické opětovné připojení klienta k demonstraci zotavení po selhání správce front. Produkt **amqsfhac** kontroluje, zda správce front používající síťově připojený úložný prostor udržuje integritu dat po selhání.

Programy **amqsghac**, **amqsphac**a **amqsmhac** se spouští z příkazového řádku a lze je použít v kombinaci k demonstraci opětovného připojení po selhání jedné instance správce front s více instancemi.

Alternativně můžete také použít ukázky **amqsghac**, **amqsphac**a **amqsmhac** k demonstraci opětovného připojení klienta k jednotlivým správcům front instance, které jsou obvykle nakonfigurované do skupiny správců front.

Chcete-li zachovat jednoduchý příklad, takže je snadné je konfigurovat, zobrazí se ukázkové programy, které se znovu připojují k jednomu správci front instance, který je spuštěn, zastaven a znovu restartován; viz “Nastavení a řízení správce front” na stránce 1070.

Použijte **amqsfhac** paralelně s **amqmfsck** , abyste zkontrolli integritu systému souborů. Další informace naleznete v tématech **amqmfsck** (kontrola systému souborů) a Ověření chování sdíleného systému souborů .

amqsphac *queueName* [*qMgrNázev*]

- **amqsphac** je aplikace IBM MQ MQI client . Do fronty vkládá posloupnost zpráv se dvěma sekunsekundové prodlevou mezi každou zprávou a zobrazuje události odeslané obslužné rutině událostí.
- K vložení zpráv do fronty se nepoužívá žádný synchronizační bod.
- V jedné skupině správců front může být provedeno opětné připojení k libovolnému správci front.

amqsgac *queueName* [*qMgrName*]

- **amqsghac** je aplikace IBM MQ MQI client . Vydá zprávy z fronty a zobrazí události odeslané obslužné rutině událostí.
- K získání zpráv z fronty se nepoužívá žádný synchronizační bod.
- V jedné skupině správců front může být provedeno opětné připojení k libovolnému správci front.

amqsmhac -s *sourceQueueNázev* -t *targetQueueNázev* [-m *qMgrNázev*] [-w *waitInterval*]

- **amqsmhac** je aplikace IBM MQ MQI client . Kopíruje zprávy z jedné fronty do jiné s výchozím intervalem čekání 15 minut od poslední zprávy přijaté před dokončením programu.
- Zprávy se kopírují v rámci synchronizačního bodu.
- Opětovná připojení lze provést pouze se stejným správcem front.

amqsfhac *QueueManagerNázev QueueName SideQueueNázev InTransactionPočet RepeatCount* (0 | 1 | 2)

- **amqsfhac** je aplikace IBM MQ MQI client . Kontroluje, zda správce front pro více instancí produktu IBM MQ používá síťově připojené úložiště, jako je například NAS nebo klastrový systém souborů, a udržuje integritu dat. Postupujte podle kroků pro spuštění produktu **amqsfhac** v části Ověření chování sdíleného systému souborů.

- Při připojování k produktu *QueueManagerName*používá volbu `MQCNO_RECONNECT_Q_MGR` . Automaticky se znovu připojí, když selže správce front.
- Načítá *InTransactionCount***RepeatCount* trvalých zpráv do *QueueName* , během které způsobí, že správce front selže v libovolném počtu. **amqsfhac** se pokaždé znovu připojí ke správci front a pokračuje. Test se ujistěte, že žádné zprávy nejsou ztraceny.
- Zprávy *InTransactionCount* jsou vloženy do každé transakce. Transakce se opakuje *RepeatCount* Počet opakování. Dojde-li k selhání v rámci transakce, produkt **amqsfhac** odvolá transakci, jakmile se produkt **amqsfhac** znovu připojí ke správci front, a znovu odešle transakci.
- Také vkládá zprávy do *SideQueueName*. Využívá *SideQueueName* ke kontrole, zda jsou všechny zprávy úspěšně potvrzeny nebo odvolány z *QueueName* . Pokud zjistí nekonzistenci, zapíše chybovou zprávu.
- Vyměnit velikost výstupního trasování z **amqsfhac** nastavením posledního parametru na `(0|1|2)`.

  0
  Nejméně výstup.

  1
  Milovný výstup.

  2
  Většina výstupů.

Konfigurace připojení klienta

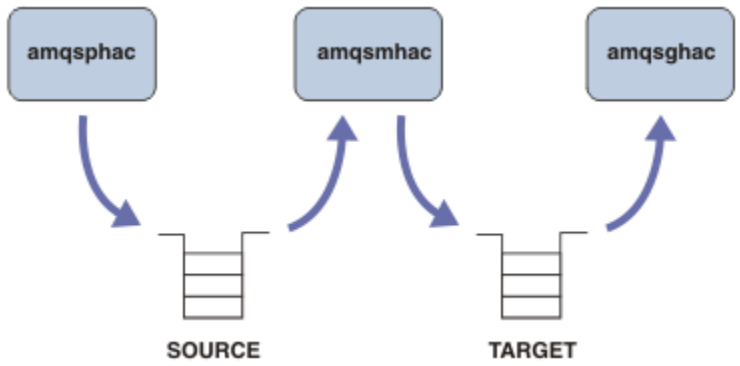
Chcete-li spustit ukázky, je třeba konfigurovat kanál připojení klienta a serveru. Procedura ověření klienta vysvětluje, jak nastavit testovací prostředí klienta.

Případně použijte konfiguraci uvedenou v následujícím příkladu.

Příklad použití amqsghac, amqsphaca amqsmhac

Příklad demonstruje opětovné připojení klientů pomocí jednoho správce front instance.

Zprávy jsou umístěny ve frontě SOURCE podle **amqsphac**, převedeny na TARGET by **amqsmhac**, a načteny z TARGET produktem **amqsghac** ; viz Obrázek 141 na stránce 1069.



Obrázek 141. Ukázky klienta pro opakované připojení

Chcete-li spustit ukázky, postupujte podle následujících kroků.

1. Vytvořte soubor `hasamples.tst` obsahující příkazy:
2. Na příkazový řádek zadejte následující příkazy:
   a. `crtmqm QM1`
   b. `strmqm QM1`
   c. `runmqsc QM1 < hasamples.tst`
3. Nastavte proměnnou prostředí **MQCHLLIB** na cestu k definičnímu souboru kanálu klienta produktu `AMQCLCHL.TAB` , například `SET MQCHLLIB=C:\IBM\MQ\MQ7\Data\qmgrs\QM1\@ipcc`.

4. Otevřete tři nová okna se sadou **MQCHLLIB** , například na Windows, zadejte třikrát **start** na předchozí příkazový řádek a spusťte každý program v jednom z oken. Viz krok “5” na stránce 1070 v příručce “Nastavení a řízení správce front” na stránce 1070.)
5. Chcete-li správce front zastavit, zadejte příkaz `endmqm -r -p QM1` a umožněte klientům připojení znovu navázat spojení.
6. Chcete-li správce front restartovat, zadejte příkaz `strmqm QM1` .

Výsledky ze spuštění ukázek **amqsghac**, **amqsphac**a **amqsmhac** na Windows jsou zobrazeny v následujících příkladech.

Nastavení a řízení správce front

1. Vytvořte správce front.

```
C:\> crtmqm QM1
IBM MQ queue manager created.
Directory 'C:\IBM\MQ\MQ7\Data\qmgrs\QM1' created.
Creating or replacing default objects for QM1.
Default objects statistics : 67 created. 0 replaced. 0 failed.
Completing setup.
Setup completed.
```

   Zapamatujte si datový adresář, abyste později nastavili proměnnou **MQCHLLIB** .
2. Spusťte správce front.

```
C:\> strmqm QM1

IBM MQ queue manager 'QM1' starting.
5 log records accessed on queue manager 'QM1' during the log replay phase.
Log replay for queue manager 'QM1' complete.
Transaction manager state recovered for queue manager 'QM1'.
IBM MQ queue manager 'QM1' started.
```

3. Vytvořte fronty a kanály, upravte port modulu listener a spusťte modul listener a kanál.
4. Zpřístupněte tabulku kanálu klienta klientům.

   Použijte datový adresář vrácený z příkazu **crtmqm** v kroku “1” na stránce 1070a přidejte do něj adresář `@ipcc` k nastavení proměnné **MQCHLLIB** .

```
C:\> SET MQCHLLIB=C:\IBM\MQ\MQ7\Data\qmgrs\QM1\@ipcc
```

5. Spuštění ukázkových programů v ostatních oknech

```
C:\> start amqsphac SOURCE QM1
C:\> start amqsmhac -s SOURCE -t TARGET -m QM1
C:\> start amqsghac TARGET QM1
```

6. Ukončete správce front a znovu jej spusťte.

```
C:\> endmqm -r -p QM1

Waiting for queue manager 'QM1' to end.
IBM MQ queue manager 'QM1' ending.
IBM MQ queue manager 'QM1' ended.


C:\> strmqm QM1

IBM MQ queue manager 'QM1' starting.
5 log records accessed on queue manager 'QM1' during the log replay phase.
Log replay for queue manager 'QM1' complete.
Transaction manager state recovered for queue manager 'QM1'.
IBM MQ queue manager 'QM1' started.
```

amqsphak

```
Sample AMQSPHAC start
target queue is SOURCE
message Message 1
message Message 2
16:25:22 : EVENT : Connection Reconnecting (Delay: 0ms)
16:25:45 : EVENT : Connection Reconnecting (Delay: 0ms)
16:26:02 : EVENT : Connection Reconnectedmessage
Message 3
message Message 4
message Message 5
```

amqsmhak

```
Sample AMQSMHA0 start
16:25:22 : EVENT : Connection Reconnecting (Delay: 0ms)
16:25:45 : EVENT : Connection Reconnecting (Delay: 0ms)
16:26:02 : EVENT : Connection Reconnected
No more messages.
Sample AMQSMHA0 end
C:\>
```

amqsgak

```
Sample AMQSGHAC start
message Message 1
message Message 2
16:25:22 : EVENT : Connection Reconnecting (Delay: 0ms)
16:25:45 : EVENT : Connection Reconnecting (Delay: 0ms)
16:26:02 : EVENT : Connection Reconnected
message Message 3
message Message 4
message Message 5
```

Související informace

Ověření chování sdíleného systému souborů

amqmfsck (kontrola systému souborů)

Ukázkové programy pro zjišťování

Dotazové ukázkové programy se dotazujeme na některé atributy fronty pomocí volání MQINQ.

Názvy těchto programů viz “Vlastnosti demonstrovány v ukázkových programech na platformách Multiplatforms” na stránce 1036 .

Tyto programy jsou určeny ke spouštění jako spouštěné programy, takže jejich jediným vstupem je struktura MQTMC2 (spouštěcí zpráva) pro systémy IBM i, Windows, UNIX and Linux . Tato struktura obsahuje název cílové fronty s atributy, které mají být zjišťovány. Verze jazyka C také používá název správce front. Verze COBOL používá výchozího správce front.

Chcete-li, aby spouštěcí proces fungoval, ujistěte se, že dotazovací program zjišťování, který chcete použít, je spuštěn zprávami přicházejícími do fronty SYSTEM.SAMPLE.INQ. Chcete-li tak učinit, zadejte název dotazovacího programu pro zjišťování, který chcete použít v poli *ApplicId* definice procesu SYSTEM.SAMPLE.INQPROCESS. IBM i Pro IBM imůžete použít příkaz CHGMQMPRC; podrobnosti najdete v tématu Změna procesu MQ (CHGMQMPRC). Ukázková fronta má typ spouštěče FIRST; pokud již ve frontě existují zprávy, než spustíte ukázku požadavku, ukázka dotazu se nespustí pomocí zpráv, které odešlete.

Pokud jste správně nastavili definici:

- ULW V případě operačního systému UNIX, Linux, and Windowsspusťte program **runmqtrm** v jedné relaci a pak spusťte program `amqsreq` v jiném.

- IBM i V případě produktu IBM ispusťte program AMQSERV4 v jedné relaci a poté spusťte program AMQSREQ4 v jiném. Můžete použít AMQSTRG4 místo AMQSERV4, ale potenciální zpoždění při odeslání úlohy by mohla méně snadno sledovat, co se děje.

Ukázkové programy požadavku slouží k odesílání zpráv požadavků, z nichž každá obsahuje pouze název fronty, do fronty SYSTEM.SAMPLE.INQ. Pro každou zprávu požadavku odesílají ukázkové programy Inquire zprávu s informacemi o frontě zadané ve zprávě požadavku. Odpovědi se posílají do fronty odpovědí uvedené ve zprávě s požadavkem.

IBM i Pokud je v produktu IBM ipoužit člen ukázkového vstupního souboru QMQMSAMP.AMQSDATA(INQ), neexistuje poslední uvedená fronta, takže ukázka vrátí zprávu se sestavou s kódem příčiny selhání.

Návrh ukázkového programu Inquire

Program otevře frontu uvedenou ve struktuře zpráv spouštěče, která byla předána, když byla spuštěna. (Pro srozumitelnost zavoláme tuto *frontu požadavků*.) Program používá volání MQOPEN k otevření této fronty pro sdílený vstup.

Program používá volání MQGET k odstranění zpráv z této fronty. Toto volání používá volby MQGMO_ACCEPT_TRUNCATED_MSG a MQGMO_WAIT, s intervalem čekání 5 sekund. Program testuje deskriptor každé zprávy za účelem zjištění, zda se jedná o zprávu požadavku; pokud není, program zahodí zprávu a zobrazí varovnou zprávu.

Pro každou zprávu požadavku odebranou z fronty požadavků program přečte název fronty (která bude nazývat *cílovou frontou*). obsažené v datech a otevření této fronty pomocí volání MQOPEN s volbou MQOO_INQ. Program potom použije volání MQINQ k dotazům na hodnoty atributů *InhibitGet*, **CurrentQDepth**a **OpenInputCount** cílové fronty.

Je-li volání MQINQ úspěšné, program použije volání MQPUT1 k vložení zprávy odpovědi do fronty pro odpovědi. Tato zpráva obsahuje hodnoty tří atributů.

Je-li volání MQOPEN nebo MQINQ neúspěšné, program použije volání MQPUT1 k vložení zprávy do fronty do fronty pro odpověď. V poli *Feedback* v deskriptoru zprávy této zprávy je kód příčiny vrácený voláním MQOPEN nebo MQINQ, v závislosti na tom, který z nich selhal.

Po volání MQINQ program zavře cílovou frontu pomocí volání MQCLOSE.

Pokud ve frontě požadavků nejsou žádné zprávy, program tuto frontu zavře a odpojí se od správce front.

The Inquire Properties of a Message Handle sample program

AMQSIQMA je ukázkový program C pro zjištění vlastností obsluhy zprávy z fronty zpráv a je příkladem použití volání rozhraní API MQINQMP.

Tato ukázka vytvoří popisovač zprávy a vloží jej do pole MsgHandle struktury MQGMO. Ukázka pak získá jednu zprávu a vypíše a vytiskne všechny vlastnosti, se kterými byl popisovač zprávy naplněn daty.

```
C:\Program Files\IBM\MQ\tools\c\Samples\Bin >amqsiqm Q QM1
Sample AMQSIQMA start
property name MyProp value MyValue
message text Hello world!
Sample AMQSIQMA end
```

Ukázkové programy publikování/odběru

Ukázkové programy publish/subscribe demonstrují použití funkcí publikování a odběru v produktu IBM MQ.

K dispozici jsou tři ukázkové programy jazyka C ilustrující, jak se mají programovat v rozhraní IBM MQ publish/subscribe. Existuje několik ukázek jazyka C, které používají starší rozhraní, a existují ukázky produktu Java . Ukázky produktu Java používají rozhraní publish/subscribe produktu IBM MQ v rozhraní com.ibm.mq.jar a rozhraní JMS publish/subscribe v com.ibm.mqjms. Ukázky JMS nejsou zahrnuty v tomto tématu.

C

Najděte ukázku vydavatele amqspub ve složce ukázek produktu C . Spusťte jej s libovolným názvem tématu, který chcete použít jako první parametr, následovaný volitelným názvem správce front. Například `amqspub mytopic QM3` . Existuje také verze klienta nazvaná amqspubc. Pokud se rozhodnete spustit verzi klienta, nejprve si prohlédněte “Konfigurace správce front pro příjem klientských připojení na platformách Multiplatforms” na stránce 1045 , kde získáte podrobnosti.

Vydavatel se připojí k výchozímu správci front a odpoví s výstupem, `target topic is mytopic` . Každý řádek, který zadáte do tohoto okna od této chvíle, bude publikován do `mytopic` .

Otevřete jiné příkazové okno ve stejném adresáři a spusťte program odběratele amqssub, dodejte jej se stejným názvem tématu a volitelným názvem správce front. Například `amqssub mytopic QM3` .

Odběratel odpovídá na výstup, `Calling MQGET : 30 seconds wait time` . Od této chvíle se řádky, které zadáte do vydavatele, objevují ve výstupu odběratele.

Spusťte jiného odběratele v jiném okně s příkazovým řádkem a sledujte, jak odběratelé přijímají publikace.

Úplnou dokumentaci parametrů, včetně nastavení voleb, naleznete v ukázkovém zdrojovém kódu. Hodnoty pro pole voleb odběratele jsou popsány v následujícím tématu: Volby (MQLONG).

Existuje další ukázka odběratele amqssbx, která nabízí další volby odběru jako přepínače příkazového řádku.

Zadejte příkaz `amqssbx -d mysub -t mytopic -k` , chcete-li vyvolat odběratele pomocí trvalých odběrů, které jsou uchovány po ukončení odběratele.

Otestujte odběr publikováním další položky s použitím vydavatele. Počkejte 30 sekund, než se odběratel ukončí. Publikujte některé další položky pod stejným tématem. Restartujte odběratele. Poslední položka publikovaná v době, kdy odběratel nebyl spuštěn, ji účastník zobrazí ihned po restartu.

odkaz C

K dispozici je další sada ukázek jazyka C, které demonstrují příkazy ve frontě. Některé z těchto ukázek byly původně odeslány jako součást sady MQ0C Supportpac. Funkce, které ukázky demonstrují, jsou plně podporovány z důvodu kompatibility.

Odrazujeme vás od použití rozhraní příkazového řádku ve frontě. Je mnohem komplexnější než rozhraní API pro publikování/odběr a neexistuje žádný přesvědčivý funkční důvod pro programování složitých příkazů zařazených do fronty. Může se však nacházet vhodnější přístup ve frontě, například proto, že rozhraní již používáte, nebo proto, že vaše programovací prostředí usnadňuje sestavování složité zprávy a volání generického volání MQPUT, místo aby se vytvářela odlišná volání MQSUB.

Další ukázky jsou umístěny v podadresáři pubsub ve složce `samples` .

V produktu Tabulka 152 na stránce 1073je uvedeno šest typů ukázek.

Tabulka 152. Kategorie starších ukázkových programů jazyka C pro publikování/odběr

Kategorie	Programy	Komentáře
RFH1	`amqssr1a.c` `amqspr1a.c`	Příklad jednoduchého publikování/odběru sestaveného pomocí zpráv ve formátu RFH1 .
RFH2	`amqssr2a.c` `amqspr2a.c`	Příklad jednoduchého publikování/odběru sestaveného pomocí zpráv ve formátu RFH2 .
Ukázky MQAI	`amqsppca.c` `amqsspca.c`	Příklad jednoduchého publikování/odběru sestaveného pomocí příkazů PCF a rozhraní příkazového řádku MQAI.

Tabulka 152. Kategorie starších ukázkových programů jazyka C pro publikování/odběr (pokračování)		
Kategorie	**Programy**	**Komentáře**
MA0C Služba výsledků pomocí RFH1	`amqsgama.c` `amqsresa.c`	Služba výsledků vytvořená pomocí záhlaví RFH1 1. Vyžaduje fronty definované v `amqsgama.tst` a `amqsresa.tst` 2. Server `amqsresa` musí být spuštěn před `amqsgama`
MA0C Služba výsledků pomocí RFH2	`amqsgr2a.c` `amqsrr2a.c`	Služba výsledků sestavená pomocí záhlaví RFH2 1. Vyžaduje fronty definované v `amqsgama.tst` a `amqsresa.tst` 2. Server `amqsresa` musí být spuštěn před `amqsgama`
Ukázka uživatelské procedury pro publikování/odběr směrování	`amqspsra.c`	Demonstruje, jak změnit místo určení fronty nebo správce front pro zprávu publikování/odběru ve výstupním rámci směrování.

Java

Ukázka Java `MQPubSubApiSample.java` kombinuje vydavatele a odběratele v jednom programu. Jeho zdrojový a kompilovaný soubor třídy se nacházejí ve složce ukázek produktu `wmqjava` .

Pokud se rozhodnete spustit v režimu klienta, nejprve si prohlédněte “Konfigurace správce front pro příjem klientských připojení na platformách Multiplatforms” na stránce 1045 , kde získáte podrobnosti.

Spusťte ukázku z příkazového řádku pomocí příkazu Java , pokud máte nakonfigurované prostředí Java . Ukázku můžete spustit také z pracovního prostoru Eclipse průzkumníka produktu IBM MQ , který má již nastavovací pracovní plochu pro programování Java .

Možná budete muset změnit některé vlastnosti ukázkového programu, abyste jej mohli spustit. To provedete tak, že poskytnete parametry do prostředí JVM, nebo upravíte zdroj.

Pokyny v produktu “Spuštění ukázky MQPubSubApiSample Java” na stránce 1074 ukazují, jak spustit ukázku z pracovního prostoru Eclipse .

Spuštění ukázky MQPubSubApiSample Java
Jak spustit příkaz MQPubSubApiSample pomocí vývojových nástrojů produktu Java z platformy Eclipse .

Než začnete

Otevřete pracovní plochu Eclipse . Vytvořte nový adresář pracovního prostoru a vyberte jej. Zavřete uvítací okno.

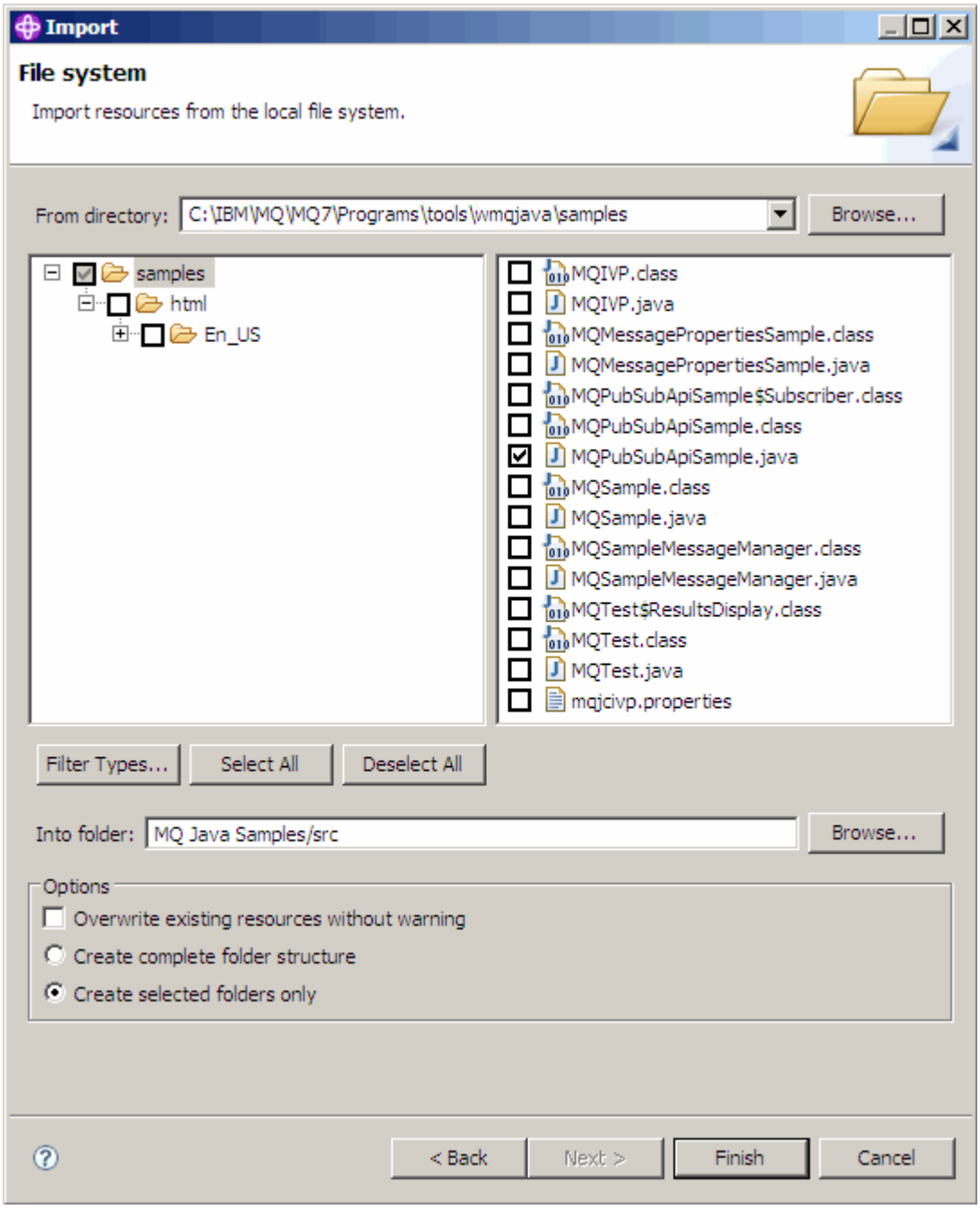
Postupujte podle kroků v produktu “Konfigurace správce front pro příjem klientských připojení na platformách Multiplatforms” na stránce 1045 před spuštěním jako klienta.

Informace o této úloze

Ukázkový program publish/subscribe Java je program IBM MQ MQI client Java . Ukázka se spustí bez úprav s použitím výchozího správce front, který naslouchá na portu 1414. Úloha popisuje tento jednoduchý případ a obecně označuje, jak poskytnout parametry, a upravit ukázku tak, aby vyhovovala různým konfiguracím IBM MQ . Příklad je ilustrován na Windows. Cesty k souborům se budou lišit i na jiných platformách.

Postup

1. Import ukázkových programů produktu Java
   a) Na pracovní ploše klepněte na nabídku **Okno** > **Otevřít perspektivu** > **Další** > **Java** a klepněte na tlačítko **OK**.
   b) Přepněte do pohledu **Průzkumník balíků** .
   c) Klepněte pravým tlačítkem myši v zobrazení **Průzkumník balíků** do seznamu povolených položek. Klepněte na volbu **Nový** > **Projekt produktuJava**.
   d) V poli **Project name** napište `MQ Java Samples`. Klepněte na tlačítko **Další**.
   e) V panelu **Java Settings** přepněte na kartu **Knihovny** .
   f) Klepněte na volbu **Přidat externí soubory JAR**.
   g) Přejděte do adresáře *MQ_INSTALLATION_PATH* `\java\lib` , kde *MQ_INSTALLATION_PATH* je instalační složka produktu IBM MQ , a vyberte `com.ibm.mq.jar` a `com.ibm.mq.jmqi.jar`
   h) Klepněte na volbu **Otevřít** > **Dokončit.**
   i) Klepněte pravým tlačítkem myši na položku `src` v zobrazení **Průzkumník balíků** .
   j) Vyberte **Importovat ...** > **Obecné** > **Systém souborů** > **Další** > **Procházet**... a přejděte k cestě *MQ_INSTALLATION_PATH* `\tools\wmqjava\samples` , kde *MQ_INSTALLATION_PATH* je instalační adresář produktu IBM MQ .
   k) Na panelu **Import** Obrázek 142 na stránce 1076klepněte na `samples` (nezaškrtne zaškrtávací políčko).
   l) Vyberte volbu `MQPubSubApiSample.java`. Pole **Into folder** by mělo obsahovat `MQ Java Samples/src`. Klepněte na tlačítko **Dokončit**.



Obrázek 142. Import systému souborů

2. Spusťte ukázkový program pro publikování/odběr.
   Existují dva způsoby spuštění programu, v závislosti na tom, zda je třeba změnit výchozí parametry.
   - První volba spustí program bez provedení jakýchkoli změn:
     - V hlavní nabídce pracovního prostoru rozbalte složku `src` . Right-click **MQPubSubApiSample.java Spustit jako** > **1. Java Aplikace**
   - Druhá volba spustí program s parametry nebo s modifikovaným zdrojovým kódem pro vaše prostředí:
     - Otevřete produkt `MQPubSubApiSample.java` a prostudujte konstruktor produktu `MQPubSubApiSample` .

– Upravte atributy programu.

  Tyto atributy jsou upravitelné pomocí přepínače -D JVM nebo zadáním výchozí hodnoty pro System propery úpravou zdrojového kódu.

  - topicObject
  - QueueManagerName
  - subscriberCount

  Tyto atributy lze měnit pouze úpravou zdrojového kódu v konstruktoru.

  - hostname
  - Port
  - kanál

  Chcete-li nastavit vlastnosti System properties, uveďte výchozí hodnotu v přístupovém objektu, například:

```
queueManagerName = System.getProperty("com.ibm.mq.pubSubSample.queueManagerName",
"QM3");
```

  Případně uveďte parametr do prostředí JVM pomocí volby `-D` , jak ukazuje následující postup:

a. Zkopírujte úplný název souboru System.Property , který chcete nastavit, například: `com.ibm.mq.pubSubSample.queueManagerName`.
b. V pracovním prostoru klepněte pravým tlačítkem myši na volbu **Spustit** > **Otevřít dialogové okno Spustit**. Poklepejte na volbu Java Aplikace v sekci **Vytvořit, spravovat a spustit aplikace** a klepněte na kartu **(x) = Argumenty** .
c. V podokně **Argumenty virtuálního počítače:** zadejte příkaz `-D` a vložte název System.property , `com.ibm.mq.pubSubSample.queueManagerName`, následovaný hodnotou `=QM3`. Klepněte na volbu **Použít** > **Spustit**.
d. Přidejte další argumenty jako seznam oddělený čárkami, nebo jako další řádky v podokně, bez oddělovačů.

   Například: `-Dcom.ibm.mq.pubSubSample.queueManagerName=QM3, -Dcom.ibm.mq.pubSubSample.subscriberCount=6`.

Ukázkový program publikování a ukončení

Příkaz AMQSPSE0 je ukázkový program jazyka C pro proceduru ukončení publikování, než je doručena odběrateli. Ukončení může například změnit záhlaví zprávy, informační obsah nebo místo určení, nebo zabránit publikování zprávy na odběrateli.

Chcete-li spustit ukázku, proveďte následující úlohy:

1. Konfigurujte správce front:
   - Linux UNIX V systémech UNIX and Linux přidejte oddíl podobný tomuto souboru `qm.ini` :

```
PublishSubscribe:
PublishExitPath=Module
PublishExitFunction=EntryPoint
```

     kde je modul *MQ_INSTALLATION_PATH*`/samp/bin/amqspse`. *MQ_INSTALLATION_PATH* představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .
   - Windows V systému Windows nastavte ekvivalentní atributy v registru.
2. Ujistěte se, že je modul přístupný pro produkt IBM MQ.
3. Restartujte správce front, aby se konfigurace sebral v konfiguraci.

4. V procesu aplikace, který má být trasován, popište, kam mají být trasovací soubory zapisovány. Příklad:
   - Linux UNIX V systému UNIX and Linux se ujistěte, že adresář `/var/mqm/trace` existuje a vyexportujte následující proměnnou prostředí:

     ```
     export MQPSE_TRACE_LOGFILE=/var/mqm/trace/PubTrace
     ```

   - Windows V systému Windowszkontrolujte, zda existuje adresář `C:\temp` a nastavte následující proměnnou prostředí:

     ```
     set MQPSE_TRACE_LOGFILE=C:\temp\PubTrace
     ```

Ukázkové programy Put

Ukázkové programy vkládání vloží zprávy do fronty pomocí volání MQPUT.

Názvy těchto programů viz “Vlastnosti demonstrovány v ukázkových programech na platformách Multiplatforms” na stránce 1036 .

Návrh ukázkového programu Put

Program používá volání MQOPEN s volbou MQOO_OUTPUT k otevření cílové fronty pro vkládání zpráv.

Pokud nemůže frontu otevřít, výstupem programu je chybová zpráva obsahující kód příčiny vrácený voláním MQOPEN. Chcete-li program ponechat jednoduchý, a to i při následných voláních MQI, program použije výchozí hodnoty pro celou řadu voleb.

Pro každý řádek vstupu program přečte text do vyrovnávací paměti a použije volání MQPUT k vytvoření datagramové zprávy obsahující text této řádky. Program pokračuje, dokud nedojde k dosažení konce vstupu nebo volání MQPUT selže. Pokud se program dostane na konec vstupu, zavře frontu pomocí volání MQCLOSE.

Spuštění ukázkových programů pro vkládání

Spuštění ukázek amqsput a amqsputc

ULW

Ukázka amqqsput je program pro vkládání zpráv s použitím lokálních vazeb a ukázkový program amqsputc je program určený pro použití vazeb klienta. Tyto programy používají následující poziční parametry:

1. Název cílové fronty (povinné)
2. Název správce front (volitelný)

   Není-li určen správce front, připojuje se hodnota amqsput k výchozímu správci front a amqsputc se připojí ke správci front určenému proměnnou prostředí MQSERVER nebo definičním souborem kanálu klienta.
3. Volby otevření (volitelné)

   Nejsou-li volby otevření zadány, použije vzorek hodnotu 8208, což je kombinace těchto dvou voleb:
   - MQOOK_VÝSTUP
   - UVÁDĚNÍ MQOO_FAIL_IF_QUIESCING
4. Volby zavření (volitelné)

   Nejsou-li volby uzavření zadány, bude v ukázce použita hodnota 0, což je MQCO_NONE.
5. Název cílového správce front (volitelné)

   Není-li správce cílové fronty zadán, bude pole `ObjectQMgrName` v MQOD ponecháno prázdné.
6. Název dynamické fronty (volitelné)

Není-li zadán název dynamické fronty, bude pole DynamicQName v MQOD ponecháno prázdné.

Tyto programy také používají proměnnou prostředí s názvem **MQSAMP_USER_ID** , která by měla být nastavena na ID uživatele, které má být použito pro ověření připojení. Je-li nastavena tato volba, program zobrazí výzvu k zadání hesla, které má být připojeno k tomuto ID uživatele.

Chcete-li spustit tyto programy, zadejte jednu z následujících možností:

- `amqsput myqueue qmanagername`
- `amqsputc myqueue qmanagername`

kde myqueue je název fronty, na kterou se mají zprávy vložit, a qmanagername je správce front, který vlastní myqueue.

Spuštění ukázky amq0put

Verze COBOL nemá žádné parametry. Připojuje se k výchozímu správci front, a když jej spustíte, budete vyzváni:

```
Please enter the name of the target queue
```

Vezme vstup z StdIn a přidá každý řádek vstupu do cílové fronty. Prázdný řádek označuje, že již nejsou žádná data.

Spuštění ukázky AMQSPUT4 C (IBM i)

Program C AMQSPUT4, který je k dispozici pouze pro platformu IBM i , vytváří zprávy čtením dat ze členu zdrojového souboru.

Při spouštění programu je třeba zadat název souboru jako parametr. Struktura souboru musí být:

```
queue name
text of message 1
text of message 2
⋮
text of message n
blank line
```

Ukázka vstupu pro vložení ukázek je dodána v knihovně QMQMSAMP souboru AMQSDATA member PUT.

Poznámka: Nezapomeňte, že názvy front jsou citlivé na velikost písmen. Všechny fronty vytvořené vzorovým souborem vytvoření AMQSAMP4 mají názvy vytvořené velkými písmeny.

Program v jazyku C vloží zprávy do fronty uvedené v prvním řádku souboru. Můžete použít dodanou frontu SYSTEM.SAMPLE.LOCAL. Program vloží text každého z následujících řádků souboru do samostatných datagramů, a zastaví se, když čte prázdný řádek na konci souboru.

Pomocí vzorového datového souboru je příkaz:

```
CALL PGM(QMQM/AMQSPUT4) PARM('QMQMSAMP/AMQSDATA(PUT)')
```

Spuštění ukázky jazyka COBOL AMQ0PUT4 (IBM i)

Program COBOL AMQ0PUT4, který je k dispozici pouze na platformě IBM i , vytváří zprávy přijímáním dat z klávesnice.

Chcete-li spustit program, zavolejte program a zadejte jméno cílové fronty jako parametr programu. Program přijímá vstup z klávesnice do vyrovnávací paměti a vytváří datagramovou zprávu pro každý řádek textu. Program se zastaví, když zadáte prázdný řádek na klávesnici.

Ukázkové programy Referenční zprávy

Ukázky referenční zprávy umožňují přenos velkého objektu z jednoho uzlu do jiného (obvykle na různých systémech) bez nutnosti uložení objektu ve frontách produktu IBM MQ ve zdrojovém nebo v cílových uzlech.

K dispozici je sada ukázkových programů, které demonstrují, jak lze do fronty vložit referenční zprávy, které jsou přijímány uživatelskými procedurami a které byly převzaty z fronty. Vzorové programy používají referenční zprávy k přesunu souborů. Chcete-li přesunout jiné objekty, jako jsou například databáze, nebo chcete-li provést kontroly zabezpečení, definujte vlastní uživatelskou proceduru na základě ukázky amqsxrm.

Verze ukázkového programu výstupního bodu zpráv, který má být použit, závisí na platformě, na které je kanál spuštěn:

- Na všech platformách použijte na odesílající straně amqsxrma.
- Použijte amqsxrma na přijímajícím konci, pokud je přijímač spuštěn pod jakoukoli platformou kromě IBM i.
- IBM i Je-li příjemce spuštěn v systému IBM i, použijte amqsxrm4.

.

IBM i *Poznámky pro uživatele produktu IBM i*

Chcete-li přijmout referenční zprávu pomocí ukázkové uživatelské procedury pro zpracování zprávy, zadejte soubor v kořenovém systému souborů IFS nebo kterýkoli podadresář tak, aby mohl být vytvořen proudový soubor.

Ukázkový uživatelská procedura pro příkaz IBM i vytvoří soubor, převede data na EBCDIC a nastaví kódovou stránku na kódovou stránku systému. Tento soubor pak můžete zkopírovat do knihovny QSYS.LIB souborový systém pomocí příkazu CPYFRMSTMF. Příklad:

```
CPYFRMSTMF FROMSTMF('JANEP/TEST.TXT')
TOMBR('qsys.lib.janep.lib/test.fie/test.mbr') MBROPT(*REPLACE)
CVTDTA(*NONE)
```

Příkaz CPYFRMSTMF nevytvořil soubor. Před spuštěním tohoto příkazu jej musíte vytvořit.

Pokud odešlete soubor z QSYS.LIB, žádné změny se nepožadují pro ukázky. Pro jakýkoli jiný systém souborů se ujistěte, že CCSID uvedený v poli CodedCharSetId ve struktuře MQRMH odpovídá hromadným datům, která odesíláte.

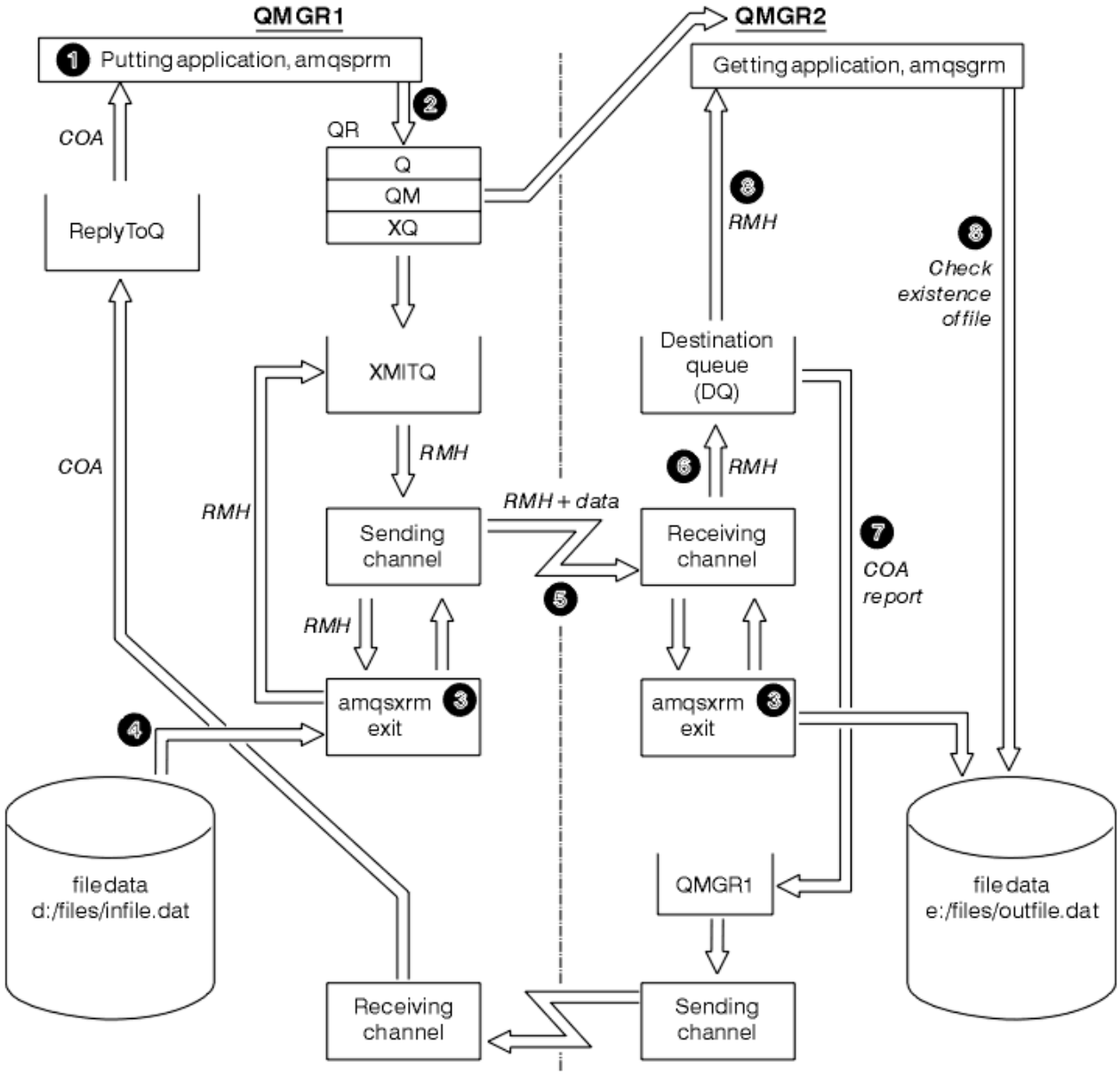
Používáte-li integrovaný systém souborů, vytvořte programové moduly pomocí volby SYSIFCOPT (*IFSIO). Chcete-li přesunout databázi nebo soubory záznamů s pevnou délkou, definujte vlastní uživatelskou proceduru na základě dodané ukázky AMQSXRM4.

Doporučeným způsobem přenosu databázového souboru je převést jej na strukturu IFS pomocí příkazu CPYTOSTMF a pak odeslat referenční zprávu připojovanou k souboru IFS. Pokud se rozhodnete přenést databázový soubor odkazem na něj z IFS, ale nekonvertovat ho na strukturu IFS, musíte uvést jméno členu. Integrita dat není zaručena, pokud zvolíte tuto metodu.

Spuštění ukázek referenční zprávy

Tento příklad použijte pro zjištění, jak spustit ukázkovou aplikaci ukázkové zprávy AMQSPRM, nebo AMQSPRMA na systému IBM i. Příklad ukazuje, jak lze do fronty vložit referenční zprávy, přijaté od ukončení zprávy a převzaty z fronty.

Ukázky referenční zprávy se spouští následujícím způsobem:



Obrázek 143. Spuštění ukázek referenční zprávy

1. Nastavte prostředí pro spuštění listenerů, kanálů a monitorů spouštěčů a definujte kanály a fronty.

   Pro účely popisu, jak nastavit referenční zprávu, se tento příklad odkazuje na odesílající stroj jako MACHINE1 se správcem front QMGR1 a přijímajícím počítačem jako MACHINE2 se správcem front s názvem QMGR2.

   Poznámka: Následující definice umožňují sestavení Referenční zprávy k odeslání souboru s typem objektu FLATFILE ze správce front QMGR1 do QMGR2 a k opětného vytvoření souboru, jak je definován ve volání do AMQSPRM (nebo AMQSPRMA na systému IBM i). Referenční zpráva (včetně dat souboru) se odešle pomocí kanálu CHL1 a přenosové fronty XMITQ a umístí se do fronty DQ. Výjimka a zprávy COA jsou odeslány zpět do QMGR1 pomocí zprávy kanálu REPORT a přenosové fronty QMGR1.

   Aplikace, která přijme referenční zprávu (AMQSGRM nebo AMQSGRMA na systému IBM i) se spouští za použití inicializační fronty INITQ a procesu PROC. Ujistěte se, že jsou pole CONNAME správně nastavena a pole MSGEXIT odráží vaši adresářovou strukturu, v závislosti na typu počítače a na tom, kde je nainstalován produkt IBM MQ .

   Definice MQSC používají styl produktu AIX pro definování uživatelských procedur, takže pokud používáte prostředí MQSC na serveru IBM i, musíte tyto definice příslušně upravit. Je důležité si

uvědomit, že data zprávy FLATFILE jsou citlivá na velikost písmen a ukázka nebude fungovat, pokud není psána velkými písmeny.

Na počítači MACHINE1, správce front QMGR1

Syntaxe MQSC

```
define chl(chl1) chltype(sdr) trptype(tcp) conname('machine2') xmitq(xmitq)
msgdata(FLATFILE) msgexit('/usr/lpp/mqm/samp/bin/amqsxrm(MsgExit)
')

define ql(xmitq) usage(xmitq)

define chl(report) chltype(rcvr) trptype(tcp) replace

define qr(qr) rname(dq) rqmname(qmgr2) xmitq(xmitq) replace
```

IBM i syntaxe příkazu

Poznámka: Nezadáte-li název správce front, systém použije výchozího správce front.

```
CRTMQMCHL CHLNAME(CHL1) CHLTYPE(*SDR) MQMNAME(QMGR1) +
            REPLACE(*YES) TRPTYPE(*TCP) +
            CONNAME('MACHINE2(60501)') TMQNAME(XMITQ) +
            MSGEXIT(QMQM/AMQSXRM4) MSGUSRDATA(FLATFILE)

CRTMQMQ   QNAME(XMITQ) QTYPE(*LCL) MQMNAME(QMGR1) +
            REPLACE(*YES) USAGE(*TMQ)

CRTMQMCHL CHLNAME(REPORT) CHLTYPE(*RCVR) +
            MQMNAME(QMGR1) REPLACE(*YES) TRPTYPE(*TCP)

CRTMQMQ   QNAME(QR) QTYPE(*RMT) MQMNAME(QMGR1) +
            REPLACE(*YES) RMTQNAME(DQ) +
            RMTMQMNAME(QMGR2) TMQNAME(XMITQ)
```

Na počítači MACHINE2, správce front QMGR2

Syntaxe MQSC

```
define chl(chl1) chltype(rcvr) trptype(tcp)
msgexit('/usr/lpp/mqm/samp/bin/amqsxrm(MsgExit)')
        msgdata(flatfile)

define chl(report) chltype(sdr) trptype(tcp) conname('MACHINE1')
        xmitq(qmgr1)

define ql(initq)

define ql(qmgr1) usage(xmitq)

define pro(proc) applicid('/usr/lpp/mqm/samp/bin/amqsgrm')

define ql(dq) initq(initq) process(proc) trigger trigtype(first)
```

Syntaxe příkazu IBM i

Poznámka: Nezadáte-li název správce front, systém použije výchozího správce front.

```
CRTMQMCHL CHLNAME(CHL1) CHLTYPE(*RCVR) MQMNAME(QMGR2) +
REPLACE(*YES) TRPTYPE(*TCP) +
MSGEXIT(QMQM/AMQSXRM4) MSGUSRDATA(FLATFILE)

CRTMQMCHL CHLNAME(REPORT) CHLTYPE(*SDR) MQMNAME(QMGR2) +
REPLACE(*YES) TRPTYPE(*TCP) +
CONNAME('MACHINE1(60500)') TMQNAME(QMGR1)

CRTMQMQ  QNAME(INITQ) QTYPE(*LCL) MQMNAME(QMGR2) +
REPLACE(*YES) USAGE(*NORMAL)

CRTMQMQ  QNAME(QMGR1) QTYPE(*LCL) MQMNAME(QMGR2) +
```

```
REPLACE(*YES) USAGE(*TMQ)

CRTMQMPRC PRCNAME(PROC) MQMNAME(QMGR2) REPLACE(*YES) +
APPID('QMQM/AMQSGRM4')

CRTMQMQ  QNAME(DQ) QTYPE(*LCL) MQMNAME(QMGR2) +
REPLACE(*YES) PRCNAME(PROC) TRGENBL(*YES) +
INITQNAME(INITQ)
```

2. Po vytvoření objektů produktu IBM MQ :
   a. Kde je to vhodné pro platformu, spusťte modul listener pro odesílající a přijímající správce front.
   b. Spuštění kanálů CHL1 a REPORT
   c. Na přijímajícím správci front spusťte monitor spouštěčů pro inicializační frontu INITQ
3. Vyvolejte ukázkový program put referenční zprávy AMQSPRM (AMQSPRMA on IBM i) z příkazového řádku pomocí následujících parametrů:

   -m
   Název lokálního správce front. Výchozí hodnota je výchozí správce front.

   -i
   Název a umístění zdrojového souboru

   -o
   Název a umístění cílového souboru

   -q
   Název fronty

   -g
   Název správce front, ve kterém je fronta definovaná v parametru -q. Tato výchozí hodnota je určena pro správce front zadaného v parametru -m.

   -t
   Typ objektu

   -w
   Interval čekání, tj. čekací doba pro výjimky a sestavy COA z přijímajícího správce front

   Chcete-li například použít ukázku s dříve definovanými objekty, použijte následující parametry:

```
-mQMGR1 -iInput File -oOutput File -qQR -tFLATFILE -w120
```

   Zvýšení čekací doby umožňuje odeslání času velkého souboru přes síť před tím, než program položení zprávy vyprší.

```
amqsprm -q QR -m QMGR1 -i d:\x\file.in -o d:\y\file.out -t FLATFILE
```

Uživatelé systému IBM i: 

   a. Zadejte následující příkaz:

```
CALL    PGM(QMQM/AMQSPRM4) PARM('-mQMGR1' +
'-i/refmsgs/rmsg1' +
'-o/refmsgs/rmsgx' '-qQR' +
'-gQMGR1' '-tFLATFILE' '-w15')
```

      Předpokládá se, že původní soubor `rmsg1` je v adresáři IFS `/refmsgs` a že chcete, aby cílový soubor byl `rmsgx` v adresáři IFS `/refmsgs` na cílovém systému.
   b. Vytvořte vlastní adresář pomocí příkazu CRTDIR, spíše než pomocí kořenového adresáře.
   c. Když zavoláte program, který vkládá data, nezapomeňte, že název výstupního souboru musí odrážet konvenci pojmenování IFS; například /TEST/FILENAME vytvoří soubor s názvem FILENAME v adresáři TEST.

Poznámka: IBM i Při zadávání parametrů můžete použít buď dopředné lomítko (/), nebo pomlčku (-).

IBM i Příklad:

```
amqsprm /i d:\files\infile.dat /o e:\files\outfile.dat /q QR
/m QMGR1 /w 30 /t FLATFILE
```

Poznámka: Pro platformy UNIX and Linux musíte použít dvě zpětná lomítka (\\) místo toho, abyste označili adresář cílového souboru. Proto příkaz **amqsprm** vypadá takto:

```
amqsprm -i /files/infile.dat -o e:\\files\\outfile.dat -q QR
-m QMGR1 -w 30 -t FLATFILE
```

Spuštění programu pro práci s referenčními zprávami provede následující akce:

- Referenční zpráva je vložena do fronty QR ve správci front QMGR1.
- Zdrojový soubor a cesta jsou `d:\files\infile.dat` a existují v systému, kde je vydán příklad příkazu.
- Je-li fronta QR vzdálenou frontou, odešle se referenční zpráva jinému správci front, na jiném systému, kde je soubor vytvořen s názvem a cestou `e:\files\outfile.dat`. Obsah tohoto souboru je stejný jako zdrojový soubor.
- amqsprm čeká 30 sekund na zprávu COA z cílového správce front.
- Typ objektu je `flatfile`, takže kanál použitý pro přesouvání zpráv z fronty QR fronty musí uvádět toto v poli *MsgData* .

4. Definujete-li kanály, vyberte uživatelskou proceduru zprávy v odesílajícím i přijímajícím rámci tak, aby byla amqsxrm.

   Windows Tato hodnota je definována v systému Windows následujícím způsobem:

   ```
   msgexit(' pathname\amqsxrm.dll(MsgExit)')
   ```

   Solaris HP-UX AIX Tento postup je definován v systémech AIX, HP-UXa Solaris následujícím způsobem:

   ```
   msgexit(' pathname/amqsxrm(MsgExit)')
   ```

   Uvedete-li název cesty, zadejte úplný název. Vynecháte-li název cesty, předpokládá se, že se program nachází v cestě zadané v souboru `qm.ini` (nebo v adresáři IBM MQ for Windows, v cestě zadané v registru).
5. Uživatelská procedura kanálu přečte záhlaví referenční zprávy a vyhledá soubor, na který odkazuje.
6. Uživatelská procedura kanálu pak může soubor rozdělit před odesláním kanálu spolu se záhlavím.

   Solaris HP-UX AIX V systémech AIX, HP-UXa Solariszměňte vlastníka skupiny cílového adresáře na hodnotu 'mqm', aby mohla vzorová uživatelská procedura pro zprávy vytvořit soubor v tomto adresáři. Také změňte oprávnění k cílovému adresáři tak, aby do ní mohli zapisovat členové skupiny mqm. Data souboru nejsou uložena ve frontách produktu IBM MQ .
7. Když je poslední segment souboru zpracováván uživatelskou procedurou příjmu zpráv, umístí se referenční zpráva do cílové fronty určené parametrem amqsprm. Je-li tato fronta spuštěna (to znamená, že definice uvádí atributy fronty **Trigger**, **InitQ**a **Process**), spustí se program určený parametrem PROC cílové fronty. Program, který má být spuštěn, musí být definován v poli `ApplId` atributu **Process** .
8. Když se referenční zpráva dostane do cílové fronty (DQ), odešle se zpráva COA zpět do aplikace pro uvedení aplikace (amqsprm).

9. Ukázka Získat referenční zprávu (Get Reference Message), amqsgrm, získává zprávy z fronty zadané ve zprávě spouštěcího impulsu a kontroluje existenci souboru.

Návrh ukázky Vložit referenční zprávu (amqsprma.c, AMQSPRM4)
Toto téma poskytuje podrobný popis ukázky Vložit referenční zprávu.

Tato ukázka vytvoří referenční zprávu, která odkazuje na soubor a vkládá ji do zadané fronty:

1. Ukázka se připojí k lokálnímu správci front pomocí příkazu MQCONN.
2. Poté se otevře (MQOPEN) modelové fronty, která se používá k přijímání zpráv sestav.
3. Ukázka sestaví referenční zprávu obsahující hodnoty vyžadované k přesunu souboru, například názvy zdrojového a cílového souboru a typ objektu. Například ukázka dodávaná s produktem IBM MQ vytvoří referenční zprávu k odeslání souboru `d:\x\file.in` z produktu `QMGR1` do produktu `QMGR2` a k opětným vytvoření souboru jako `d:\y\file.out` s použitím následujících parametrů:

   ```
   amqsprm -q QR -m QMGR1 -i d:\x\file.in -o d:\y\file.out -t FLATFILE
   ```

   Kde `QR` je definice vzdálené fronty, která odkazuje na cílovou frontu v systému `QMGR2`.

   Poznámka: Na platformách UNIX and Linux použijte místo jednoho k označení adresáře cílového souboru dvě zpětná lomítka (\\) místo jednoho. Proto příkaz **amqsprm** vypadá takto:

   ```
   amqsprm -q QR -m QMGR1 -i /x/file.in -o d:\\y\\file.out -t FLATFILE
   ```

4. Referenční zpráva je vložena (bez jakýchkoliv dat souboru) do fronty zadané parametrem /q. Jedná-li se o vzdálenou frontu, je zpráva vložena do příslušné přenosové fronty.
5. Ukázka čeká, po dobu uvedenou v parametru /w (což je výchozí hodnota 15 sekund), pro sestavy COA, které jsou spolu s hlášeními výjimek odeslány zpět do dynamické fronty vytvořené v lokálním správci front (QMGR1).

Návrh ukázky ukončení referenční zprávy (amqsxrma.c, AMQSXRM4)
Tato ukázka rozpoznává referenční zprávy s typem objektu, který odpovídá typu objektu v poli uživatelských dat uživatelské procedury pro zpracování zprávy definice kanálu.

Pro tyto zprávy nastane následující situace:

- Na odesílacím nebo serverovém kanálu se uvedená délka dat zkopíruje z uvedeného posunutí uvedeného souboru do prostoru zbývajícího ve vyrovnávací paměti agenta po referenční zprávě. Není-li dosaženo konce souboru, vrátí se referenční zpráva po aktualizaci pole *DataLogicalOffset* do přenosové fronty.
- V případě žadatele nebo kanálu příjemce, je-li pole *DataLogicalOffset* nula a uvedený soubor neexistuje, je vytvořen. Data následující za referenční zprávou se přidají na konec uvedeného souboru. Není-li referenční zpráva poslední pro uvedený soubor, bude vyřazena. Jinak se vrátí do uživatelské procedury kanálu bez připojených dat, které mají být vloženy do cílové fronty.

Pokud je pro odesílací a serverové kanály pole *DataLogicalLength* ve vstupní referenční zprávě nula, zbývající část souboru, od *DataLogicalOffset* do konce souboru, se má odeslat spolu s kanálem. Pokud není nula, odešle se pouze uvedená délka.

Dojde-li k chybě (například v případě, že ukázka nemůže otevřít soubor), MQCXP. *ExitResponse* je nastaven na MQXCC_SUPPRESS_FUNCTION, aby zpracovávaná zpráva byla vložena do fronty nedoručených zpráv místo toho, aby pokračovala v cílové frontě. Kód zpětné vazby je vrácen v MQCXP. *Feedback* a vrátí se do aplikace, která vložila zprávu do pole *Feedback* v deskriptoru zpráv ve zprávě sestavy. Důvodem je to, že vložení aplikace požadovala výjimku nastavením MQRO_EXCEPTION v poli *Report* MQMD.

Je-li kódování nebo *CodedCharacterSetId* (CCSID) referenční zprávy odlišné od kódování ve správci front, je referenční zpráva převedena na lokální kódování a CCSID. V našem vzorku, amqsprm, formát objektu je MQFMT_STRING, takže amqsxrm převádí data objektu na lokální CCSID na přijímajícím konci před tím, než jsou data zapsána do souboru.

Neuvádějte formát přenášený soubor jako MQFMT_STRING, pokud soubor obsahuje vícebajtové znaky (například DBCS nebo Unicode). Důvodem je skutečnost, že vícebajtový znak lze rozdělit, je-li soubor segmentován na konci odesílání. Chcete-li přenést a převést takový soubor, zadejte formát jako něco jiného než MQFMT_STRING tak, aby se jeho uživatelská procedura pro referenční zprávy nekonvertoval a nepřeváděla soubor na přijímajícím konci po dokončení přenosu.

Kompilace ukázky Ukončení referenční zprávy
Chcete-li zkompilovat ukázku Výstup referenční zprávy, použijte příkaz pro platformu, na které je nainstalován produkt IBM MQ .

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Chcete-li kompilovat amqsxrma, použijte následující příkazy:

zapAIX

AIX

```
xlc_r -q64 -e MsgExit -bE:amqsxrm.exp -bM:SRE -o amqsxrm_64_r
-I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib64 -lmqm_r amqsqrma.c
```

zapHP-UX

HP-UX

```
$ c89 +DD64 +z -c -D_HPUX_SOURCE -o amqsxrma.o amqsqrma.c -I MQ_INSTALLATION_PATH/inc
$ ld -b amqsxrma.o -o /var/mqm/exits64/amqsxrma -L MQ_INSTALLATION_PATH/lib64
-L/usr/lib/pa20_64 -lmqm_r -lpthread
```

zapIBM i

IBM i

```
CRTCMOD MODULE(MYLIB/AMQSXRMA) SRCFILE(QMQMSAMP/QCSRC)
TERASPACE(*YES *TSIFC)
```

Poznámka:

1. Chcete-li vytvořit váš modul tak, aby používal systém souborů IFS, přidejte volbu SYSIFCOPT(*IFSIO)
2. Chcete-li vytvořit program pro použití s kanály bez podprocesů, použijte následující příkaz: CRTPGM PGM(MYLIB/AMQSXRMA) BNDSRVPGM(QMQM/LIBMQM)
3. Chcete-li vytvořit program pro použití s kanály s podporou podprocesů, použijte následující příkaz: CRTPGM PGM(MYLIB/AMQSXRMA) BNDSRVPGM(QMQM/LIBMQM_R)

zapLinux

Linux

```
$ gcc -m64 -shared -fPIC -o /var/mqm/exits64/amqsxrma amqsqrma.c -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -Wl,-rpath= MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-lmqm_r
```

zapSolaris

Solaris

```
$ cc -xarch=v9 -mt -G -o /var/mqm/exits64/amqsxrma amqsqrma.c -I  MQ_INSTALLATION_PATH/inc
-L  MQ_INSTALLATION_PATH/lib64 -R  MQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -lmqm


-lsocket
-lnsl -ldl
```

zapWindows

Windows Produkt IBM MQ nyní dodává knihovnu mqm s klientskými balíky a s balíky serveru, takže následující příklad používá produkt `mqm.lib` namísto produktu `mqmvx.lib`:

```
cl amqsqrma.c /link /out:amqsxrm.dll /dll mqm.lib mqm.lib /def:amqsxrm.def
```

Související pojmy

“Psaní programů výstupních bodů kanálu” na stránce 929
Můžete použít následující informace, které vám pomohou psát programy výstupního bodu kanálu.

Návrh ukázky Get Reference Message (amqsgrma.c, AMQSGRM4)
Toto téma vysvětluje návrh ukázky Získat referenční zprávu.

Logika programu je následující:

1. Ukázka je spuštěna a extrahuje názvy front a správců front ze vstupní zprávy spouštěče.
2. Poté se připojí k zadanému správci front pomocí příkazu MQCONN a otevře určenou frontu pomocí příkazu MQOPEN.
3. Ukázka vydá příkaz MQGET s intervalem čekání 15 sekund uvnitř cyklu pro získání zpráv z fronty.
4. Je-li zpráva referenční zprávou, ukázka zkontroluje existenci souboru, který byl přenesen.
5. Poté frontu zavře a odpojí se od správce front.

Ukázkové programy požadavku

Vzorové programy požadavku demonstrují zpracování typu klient/server. Ukázky jsou klienti, kteří vloží zprávy požadavků do cílové fronty serveru, která je zpracována programem serveru. Čekají na program serveru, aby umístili zprávu odpovědi do fronty pro odpověď.

Ukázky požadavků umístili řadu zpráv požadavků do fronty cílového serveru pomocí volání MQPUT. Tyto zprávy určují lokální frontu SYSTEM.SAMPLE.REPLY jako fronta odpovědí, která může být lokální nebo vzdálená fronta. Programy čekají na zprávy odpovědí a pak je zobrazí. Odpovědi se odesílají pouze v případě, že je fronta cílových serverů zpracovávána serverovou aplikací nebo pokud je spuštěna aplikace pro tento účel (jsou navrženy vzorové programy Inquire, Set a Echo). Ukázka C čeká 1 minutu (ukázka COBOL čeká 5 minut), pro první odpověď na příchod (aby umožnil spuštění serverové aplikace) a 15 sekund pro následné odpovědi, ale oba vzorky mohou skončit, aniž by byly obdrženy odpovědi. Názvy ukázkových programů požadavku viz “Vlastnosti demonstrovány v ukázkových programech na platformách Multiplatforms” na stránce 1036 .

Spuštění ukázkových programů požadavku

Spuštění ukázek amqsreq0.c, amqsreq a amqsreqc

Verze C programu má tři parametry:

1. Název fronty cílového serveru (nezbytné)
2. Název správce front (volitelný)
3. Fronta odpovědí (volitelná)

Zadejte například jednu z následujících možností:

- `amqsreq myqueue qmanagername replyqueue`
- `amqsreqc myqueue qmanagername`
- `amq0req0 myqueue`

kde `myqueue` je název fronty cílového serveru, `qmanagername` je název správce front, který vlastní `myqueue`, a `replyqueue` je jméno fronty odpovědí.

Pokud název správce front vynecháte, předpokládá se, že výchozí správce front vlastní tuto frontu. Pokud vynecháte název fronty odpovědí, je poskytnuta výchozí fronta odpovědí.

Spuštění ukázky amq0req0.cbl

Verze COBOL nemá žádné parametry. Připojuje se k výchozímu správci front, a když jej spustíte, budete vyzváni:

```
Please enter the name of the target server queue
```

Program vezme svůj vstup z StdIn a přidá každý řádek do cílové fronty serveru, přičemž každý řádek textu bude mít obsah jako obsah zprávy požadavku. Program skončí, když se čte řádek s hodnotou null.

Spuštění ukázky AMQSREQ4

Program v jazyku C vytváří zprávy tak, že přebírá data ze stdin (klávesnice) s prázdným časem ukončujícím vstup. Program přijímá až tři parametry: název cílové fronty (povinné), název správce front (volitelné) a název fronty pro odpovědi (volitelné). Není-li zadán žádný název správce front, použije se výchozí správce front. Není-li zadána žádná fronta pro odpověď, je uveden parametr SYSTEM.SAMPLE.REPLY se používá.

Zde je příklad, jak volat vzorový program v jazyce C, který uvádí frontu pro odpověď, ale nechání výchozí nastavení správce front:

```
CALL PGM(QMQM/AMQSREQ4) PARM('SYSTEM.SAMPLE.LOCAL' '' 'SYSTEM.SAMPLE.REPLY')
```

Poznámka: Nezapomeňte, že názvy front jsou citlivé na velikost písmen. Všechny fronty vytvořené vzorovým souborem vytvoření AMQSAMP4 mají názvy vytvořené velkými písmeny.

Spuštění ukázky AMQ0REQ4

Program v jazyce COBOL vytváří zprávy tak, že přijímá data z klávesnice. Chcete-li spustit program, zavolejte program a uveďte název cílové fronty jako parametr. Program přijímá vstup z klávesnice do vyrovnávací paměti a vytváří zprávu požadavku pro každý řádek textu. Program se zastaví, když zadáte prázdný řádek na klávesnici.

Spuštění ukázky požadavku pomocí spouštěče

Pokud se ukázka používá se spouštěním a jedním z ukázkových programů Inquire, Set nebo Echo, musí být řádek vstupu název fronty fronty, ke které má spuštěný program přistupovat.

ULW *Spuštění ukázky požadavku pomocí spouštěče v produktu UNIX, Linux, and Windows*
V operačním systému UNIX, Linux, and Windowsspusťte program pro monitorování spouštěčů RUNMQTRM v jedné relaci a pak spusťte program amqsreq v jiné relaci.

Spuštění ukázek pomocí spouštěče:

1. Spusťte program pro monitorování spouštěčů RUNMQTRM v jedné relaci (inicializační frontu SYSTEM.SAMPLE.TRIGGER je k dispozici pro použití).
2. Spusťte program amqsreq v jiné relaci.

3. Ujistěte se, že jste definovali cílovou frontu serveru.

   K dispozici jsou ukázkové fronty, které můžete použít jako frontu cílového serveru pro ukázku požadavku na vložení zpráv:

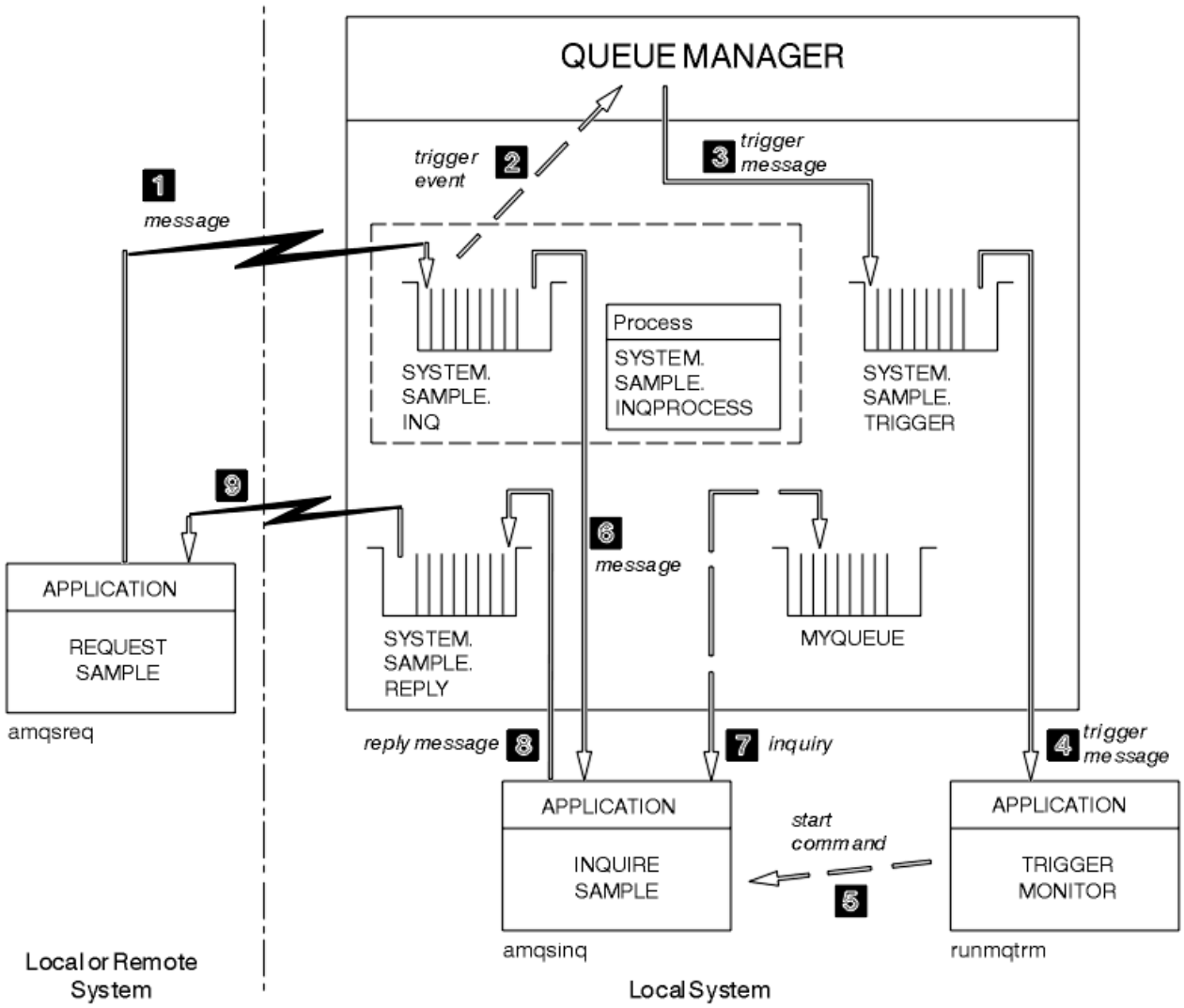
   - SYSTEM.SAMPLE.INQ -pro ukázkový program Inquire
   - SYSTEM.SAMPLE.SET -pro ukázkový program Set
   - SYSTEM.SAMPLE.ECHO -pro ukázkový program Echo

   Tyto fronty mají typ spouštěče FIRST, takže pokud již ve frontě existují zprávy před spuštěním ukázky Požadavek, serverové aplikace se nespustí pomocí zpráv, které odešlete.
4. Ujistěte se, že jste definovali frontu pro ukázkový program Inquire, Set nebo Echo, který se má použít.

To znamená, že je monitor spouštěčů připraven, když ukázka požadavku odešle zprávu.

Poznámka: Ukázkové definice procesu vytvořené pomocí RUNMQSC a amqscos0.tst spustí ukázky jazyka C. Změňte definice procesu v souboru amqscos0.tst a použijte RUNMQSC s tímto aktualizovaným souborem pro použití verzí jazyka COBOL.

Obrázek 144 na stránce 1089 demonstruje, jak používat ukázky Požadavek a Dotázat se dohromady.



Obrázek 144. Požadavek a zjišťování ukázek pomocí spouštěče

Do pole Obrázek 144 na stránce 1089 Vzorek požadavku vloží zprávy do cílové fronty serveru SYSTEM.SAMPLE.INQa Dotaz na ukázku se dotazuje fronty, MYQUEUE. Případně můžete použít jednu

z ukázkových front definovaných při spuštění příkazu amqscos0.tstnebo jakékoli jiné fronty, kterou jste definovali, pro ukázku Inquire.

Poznámka: Čísla v produktu Obrázek 144 na stránce 1089 zobrazují posloupnost událostí.

Chcete-li spustit ukázky Požadavek a odběr, použijte spuštění:

1. Zkontrolujte, zda jsou definované fronty, které chcete použít, definovány. Spusťte amqscos0.tst, abyste definovali ukázkové fronty, a definujte frontu MYQUEUE.
2. Spusťte příkaz RUNMQTRM monitoru spouštěčů:

   ```
   RUNMQTRM -m qmanagername -q SYSTEM.SAMPLE.TRIGGER
   ```

3. Spustit ukázku požadavku

   ```
   amqsreq SYSTEM.SAMPLE.INQ
   ```

   Poznámka: Objekt procesu definuje, co se má spustit. Pokud klient a server nejsou spuštění na stejné platformě, musí všechny procesy spuštěné monitorem spouštěčů definovat *ApplType*, jinak server převezme své výchozí definice (tj. typ aplikace, která je obvykle přidružena k počítači serveru) a způsobí selhání.

   Seznam typů aplikací viz ApplType.
4. Zadejte název fronty, kterou chcete použít jako vzorek pro zjišťování:

   ```
   MYQUEUE
   ```

5. Zadejte prázdný řádek (chcete-li ukončit Požadavek na program).
6. Ukázka požadavku pak zobrazí zprávu obsahující data, která program Inquire získal ze MYQUEUE.

Můžete použít více než jednu frontu; v tomto případě zadejte názvy ostatních front v kroku “4” na stránce 1090.

Další informace o spouštění viz “Spuštění aplikací produktu IBM MQ pomocí spouštěčů” na stránce 829.

IBM i *Spuštění ukázky požadavku pomocí spouštěče v produktu IBM i*

V systému IBM ispusťte ukázkový spouštěcí server AMQSERV4v jedné úloze a poté spusťte příkaz AMQSREQ4 v jiné úloze. To znamená, že spouštěcí server je připraven, když vzorový program Požadavek odešle zprávu.

Poznámka:

1. Ukázkové definice vytvořené pomocí příkazu AMQSAMP4 spouští verze jazyka C ukázek. Chcete-li spouštět verze jazyka COBOL, změňte definice procesu SYSTEM.SAMPLE.ECHOPROCESS, SYSTEM.SAMPLE.INQPROCESSa SYSTEM.SAMPLE.SETPROCESS. Můžete použít příkaz CHGMQMPRC (pro podrobnosti viz Změna procesu MQ (CHGMQMPRC)) chcete-li to provést, nebo můžete upravit a spustit vlastní verzi produktu AMQSAMP4.
2. Zdrojový kód pro AMQSERV4 se dodává pouze pro jazyk C. Avšak kompilovaná verze (kterou můžete použít s ukázkami jazyka COBOL) je dodávána v knihovně QMQM.

Do těchto ukázkových front serveru můžete vložit zprávy vzniklé při zpracování požadavku:

- SYSTEM.SAMPLE.ECHO (pro ukázkové programy Echo)
- SYSTEM.SAMPLE.INQ (pro ukázkové programy pro zjišťování)
- SYSTEM.SAMPLE.SET (pro sadu ukázkových programů)

Graf toku pro SYSTEM.SAMPLE.ECHO je zobrazen v souboru Obrázek 145 na stránce 1092. Pomocí vzorového datového souboru, příkaz pro vydání požadavku na program C pro tento server:

```
CALL PGM(QMQMSAMP/AMQSREQ4) PARM('QMQMSAMP/AMQSDATA(ECHO)')
```

Poznámka: Tato ukázková fronta má typ spouštěče FIRST, takže pokud již ve frontě existují zprávy, než spustíte ukázku Požadavek, serverové aplikace se nespustí pomocí zpráv, které odešlete.

Pokud se chcete pokusit o další příklady, můžete zkusit následující varianty:

- Použijte AMQSTRG4 (nebo jeho ekvivalent příkazového řádku STRMQMTRM, pro podrobnosti viz téma Spuštění monitoru spouštěčů MQ (STRMQMTRM)) místo AMQSERV4 místo toho odešle úlohu, ale potenciální zpoždění při odeslání úlohy by mohla méně snadno sledovat, co se děje.
- Spusťte SYSTEM.SAMPLE.INQUIRE a SYSTEM.SAMPLE.SET . Při použití vzorového datového souboru příkazy k vydání požadavků programu C na tyto servery jsou:

  ```
  CALL PGM(QMQMSAMP/AMQSREQ4) PARM('QMQMSAMP/AMQSDATA(INQ)')
  CALL PGM(QMQMSAMP/AMQSREQ4) PARM('QMQMSAMP/AMQSDATA(SET)')
  ```

  Tyto ukázkové fronty mají také typ spouštěče FIRST.

Návrh ukázkového programu Požadavek
Program otevře cílovou frontu serveru tak, aby mohla vkládat zprávy. Používá volání MQOPEN s volbou MQOO_OUTPUT. Nemůže-li frontu otevřít, zobrazí se v programu chybová zpráva obsahující kód příčiny vrácený voláním MQOPEN.

Program pak otevře frontu pro odpověď s názvem SYSTEM.SAMPLE.REPLY , aby bylo možné získat zprávy odpovědí. Za tímto způsobem program používá volání MQOPEN s volbou MQOO_INPUT_EXCLUSIVE. Nemůže-li frontu otevřít, zobrazí se v programu chybová zpráva obsahující kód příčiny vrácený voláním MQOPEN.

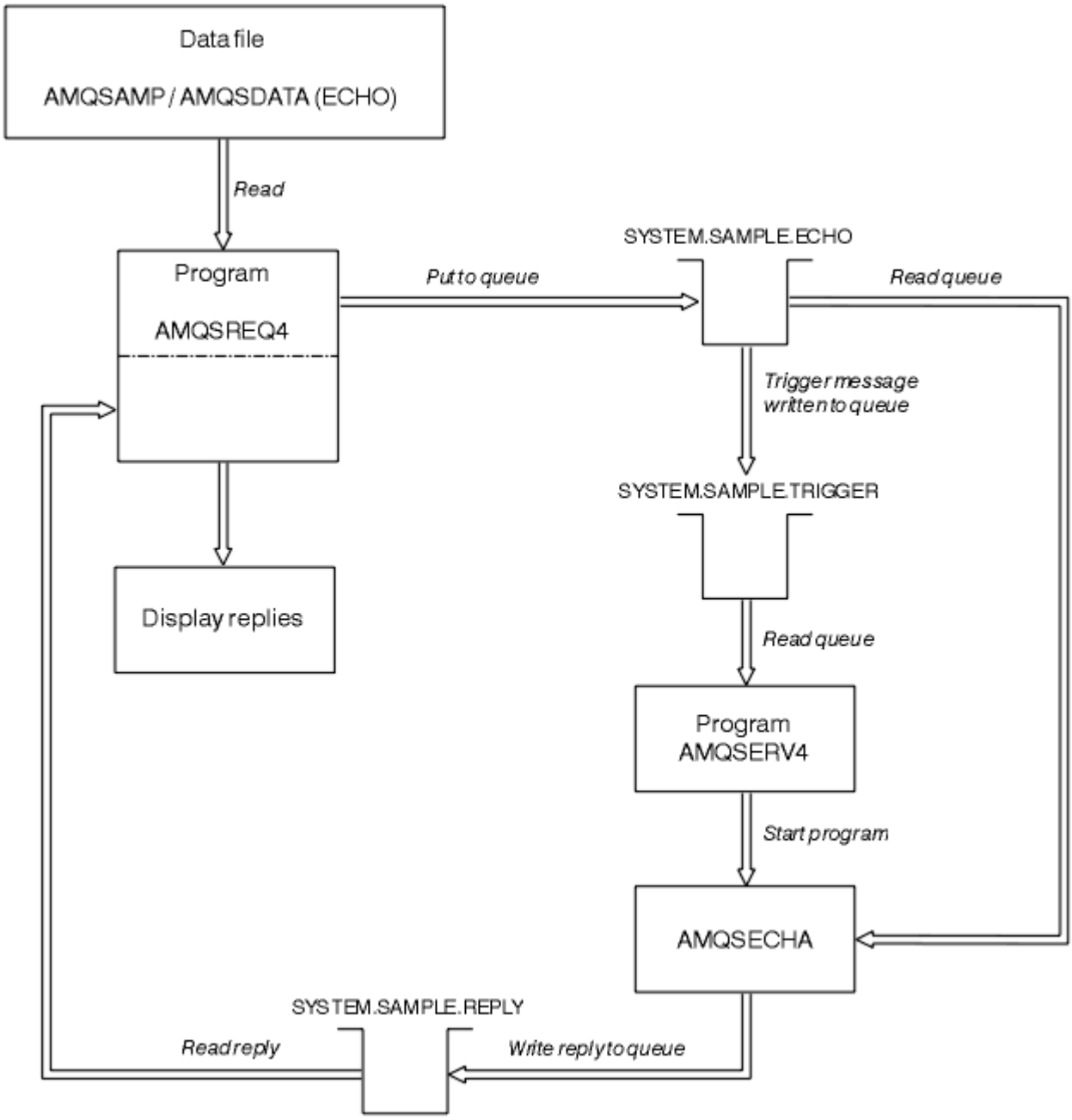
Pro každý řádek vstupu program pak přečte text do vyrovnávací paměti a použije volání MQPUT k vytvoření zprávy požadavku obsahující text této řádky. Při tomto volání program používá volbu sestavy MQRO_EXCEPTION_WITH_DATA, aby bylo možné požádat, aby všechny zprávy sestavy odeslané o zprávě požadavku obsahovaly prvních 100 bajtů dat zprávy. Program pokračuje, dokud nedojde k dosažení konce vstupu nebo volání MQPUT selže.

Program pak použije volání MQGET k odstranění zpráv odpovědí z fronty a zobrazí data obsažená v odpovědích. Volání MQGET používá volby MQGMO_WAIT, MQGMO_CONVERT a MQGMO_ACCEPT_TRUNCATED. *WaitInterval* je 5 minut ve verzi jazyka COBOL a 1 minutu ve verzi C, pro první odpověď (k povolení času pro aplikaci serveru) a 15 sekund pro následné odpovědi. Program čeká na tato období, pokud ve frontě není žádná zpráva. Pokud před uplynutím tohoto intervalu nepřijde žádná zpráva, volání se nezdaří a vrátí kód příčiny MQRC_NO_MSG_AVAILABLE. Volání také používá volbu MQGMO_ACCEPT_TRUNCATED_MSG, takže zprávy, které jsou delší než deklarovaná velikost vyrovnávací paměti, jsou oříznuty.

Tento program ukazuje, jak vymazat pole *MsgId* a *CorrelId* struktury MQMD po každém volání MQGET, protože volání nastavuje tato pole na hodnoty obsažené ve zprávě, kterou načítá. Vymazání těchto polí znamená, že po sobě jdoucí příkazy MQGET načítají zprávy v pořadí, ve kterém jsou zprávy zadrženy ve frontě.

Program pokračuje, dokud buď volání MQGET nevrátí kód příčiny MQRC_NO_MSG_AVAILABLE, nebo se volání MQGET nezdaří. Pokud se volání nezdaří, zobrazí program chybovou zprávu, která obsahuje kód příčiny.

Program pak zavře jak frontu cílového serveru, tak frontu pro odpověď pomocí volání MQCLOSE.



Obrázek 145. Ukázkový graf vývojového diagramu IBM i Klient/Server (Echo)

Ukázkové programy Set

Ukázkové programy brání vložení operací do fronty pomocí volání MQSET za účelem změny atributu **InhibitPut** fronty. Také se dozvíte o návrhu ukázkových programů.

Názvy těchto programů viz “Vlastnosti demonstrovány v ukázkových programech na platformách Multiplatforms” na stránce 1036 .

Programy jsou určeny ke spuštění jako spuštěné programy, takže jejich jediným vstupem je struktura MQTMC2 (zpráva spouštěče), která obsahuje název cílové fronty s atributy, které mají být dotazovány. Verze jazyka C také používá název správce front. Verze COBOL používá výchozího správce front.

Aby spouštěcí proces fungoval, ujistěte se, že ukázkový program Nastavení, který chcete použít, je spuštěn zprávami přicházejícími do fronty SYSTEM.SAMPLE.SET. Chcete-li to provést, zadejte název ukázkového programu Set, který chcete použít v poli *ApplicId* definice procesu

SYSTEM.SAMPLE.SETPROCESS. Ukázková fronta má typ spouštěče FIRST; pokud již ve frontě existují zprávy, než spustíte ukázku Požadavek, nespustí se ukázka Nastavit zprávy, které jste odeslali.

Pokud jste správně nastavili definici:

- 
Pro systémy UNIX, Linux, and Windows spusťte program **runmqtrm** v jedné relaci a pak spusťte program amqsreq v jiném.
- 
V případě produktu IBM ispusťte program AMQSERV4 v jedné relaci a poté spusťte program AMQSREQ4 v jiném. Můžete použít AMQSTRG4 místo AMQSERV4, ale potenciální zpoždění při odeslání úlohy by mohla méně snadno sledovat, co se děje.

Ukázkové programy požadavku slouží k odeslání zpráv požadavků, z nichž každá obsahuje pouze název fronty, do fronty SYSTEM.SAMPLE.SET. Pro každou zprávu požadavku odešlete ukázkové programy odeslání zprávy odpovědi obsahující potvrzení, že operace put byly na zadané frontě zablokovány. Odpovědi se posílají do fronty odpovědí uvedené ve zprávě s požadavkem.

Návrh ukázkového programu Set

Program otevře frontu uvedenou ve struktuře zpráv spouštěče, která byla předána, když byla spuštěna. (Pro srozumitelnost zavoláme tuto *frontu požadavků*.) Program používá volání MQOPEN k otevření této fronty pro sdílený vstup.

Program používá volání MQGET k odstranění zpráv z této fronty. Toto volání používá volby MQGMO_ACCEPT_TRUNCATED_MSG a MQGMO_WAIT, s intervalem čekání 5 sekund. Program testuje deskriptor každé zprávy za účelem zjištění, zda se jedná o zprávu požadavku; pokud není, program zahodí zprávu a zobrazí varovnou zprávu.

Pro každou zprávu požadavku odebranou z fronty požadavků program přečte název fronty (která bude nazývat *cílovou frontou*). obsažené v datech a otevření této fronty pomocí volání MQOPEN s volbou MQOO_SET. Program potom použije volání MQSET k nastavení hodnoty atributu **InhibitPut** cílové fronty na hodnotu MQQA_PUT_INHIBITED.

Je-li volání MQSET úspěšné, program použije volání MQPUT1 k vložení zprávy odpovědi do fronty pro odpovědi. Tato zpráva obsahuje řetězec `PUT inhibited`.

Je-li volání MQOPEN nebo MQSET neúspěšné, program použije volání MQPUT1 k vložení zprávy produktu `report` do fronty pro odpovědi. V poli *Feedback* v deskriptoru zprávy této zprávy je kód příčiny vrácený voláním MQOPEN nebo MQSET, v závislosti na tom, který z nich selhal.

Po volání MQSET program zavře cílovou frontu pomocí volání MQCLOSE.

Pokud ve frontě požadavků nejsou žádné zprávy, program tuto frontu zavře a odpojí se od správce front.

Ukázkový program TLS

AMQSSLC je ukázkový program C, který ukazuje, jak používat struktury MQCNO a MQSCO k poskytnutí informací o připojení klienta TLS v rámci volání MQCONNX. To umožňuje aplikaci klienta MQI poskytnout definici nastavení kanálu pro připojení klienta a TLS v běhovém prostředí bez tabulky definic kanálů klienta (CCDT).

Je-li zadán název připojení, vytvoří program definici kanálu připojení klienta ve struktuře MQCD.

Je-li zadán název kmene souboru úložiště klíčů, vytvoří program strukturu MQSCO; pokud je také dodána adresa URL odpovídacího modulu OCSP, program vytvoří strukturu MQAIR záznamu ověřovacích informací.

Program se poté připojí ke správci front pomocí příkazu MQCONNX. Rozvodí a vytiskne název správce front, ke kterému je připojen.

Tento program je určen k propojení s aplikací klienta MQI. Lze ji však propojit jako běžnou aplikaci MQI. Pak se jednoduše připojí k lokálnímu správci front a bude ignorovat informace o připojení klienta.

AMQSSLC přijímá následující parametry, všechny jsou volitelné:

-m QmgrName
Název správce front, ke kterému se má připojit

-c ChannelName
Název kanálu, který má být použit

-x ConnName
Název připojení serveru

Parametry TLS:

-k KeyReposStem
Název kmene souboru úložiště klíčů. Jedná se o úplnou cestu k souboru bez přípony .kdb. Příklad:

```
/home/user/client
C:\User\client
```

-s CipherSpec
Řetězec CipherSpec kanálu TLS odpovídající hodnotě SSLCIPH v definici kanálu SVRCONN ve správci front.

-f
Uvádí, že musí být použity pouze certifikované algoritmy FIPS 140-2.

-b VALUE1[,VALUE2...]
Určuje, že musí být použity pouze algoritmy standardu Suite B. Tento parametr je seznam s čárkami jako oddělovači jedné nebo více následujících hodnot: NONE,128_BIT,192_BIT. Tyto hodnoty mají stejný význam jako hodnoty proměnné prostředí MQSUITEB a ekvivalentní nastavení EncryptionPolicySuiteB v sekci konfiguračního souboru klienta zabezpečení SSL.

-p Zásada
Uvádí zásadu ověření certifikátu, která se má použít. Může jít o jednu z následujících hodnot:

ANY
Použít všechny zásady ověření platnosti certifikátů podporované knihovnou zabezpečených soketů a přijmout řetěz certifikátů, pokud některý ze zásad považuje řetěz certifikátů za platný. Toto nastavení lze použít pro maximální zpětnou kompatibilitu se staršími digitálními certifikáty, které nesplňují moderní certifikační standardy.

RFC5280
Použít pouze zásadu ověření platnosti certifikátu vyhovujícího RFC 5280. Toto nastavení poskytuje přísnější validaci než nastavení ANY, ale odmítá některé starší digitální certifikáty.

Výchozí hodnota je ANY.

Parametr odvolání certifikátu OCSP:

-o adresa URL
Adresa URL odpovídacího modulu OCSP

Spuštění ukázkového programu TLS
Chcete-li spustit ukázkový program TLS, musíte nejprve nastavit své prostředí TLS. Poté spustíte ukázku z příkazového řádku a dodáte tak počet parametrů.

Informace o této úloze

Následující pokyny spouštějí ukázkový program s použitím osobních certifikátů. Tímto příkazem můžete například pomocí certifikátu CA používat certifikáty certifikačních autorit a kontrolovat jejich stav pomocí odpovídacího modulu OCSP. Viz pokyny v rámci ukázky.

Postup

1. Vytvořte správce front s názvem QM1. Další informace viz crtmqm.
2. Vytvořte úložiště klíčů pro správce front. Další informace naleznete v tématu Nastavení úložiště klíčů v systému UNIX, Linux, and Windows.

3. Vytvořte úložiště klíčů pro klienta. Nazvěte jej *clientkey.kdb*.
4. Vytvořte osobní certifikát pro správce front. Další informace viz téma Vytvoření osobního certifikátu s automatickým podpisem na serveru UNIX, Linux, and Windows.
5. Vytvořte osobní certifikát pro klienta.
6. Extrahujte osobní certifikát z úložiště klíčů serveru a přidejte jej do úložiště klienta. Další informace naleznete v tématu Extrakce veřejné části certifikátu podepsaného držitelem z úložiště klíčů v systému UNIX, Linux, and Windowsa Přidání certifikátu CA (nebo veřejné části certifikátu podepsaného držitelem) do úložiště klíčů v systémech UNIX, Linux nebo Windows.
7. Extrahujte osobní certifikát z úložiště klíčů klienta a přidejte jej do úložiště klíčů serveru.
8. Vytvořte kanál připojení k serveru pomocí příkazu MQSC:

```
DEFINE CHANNEL(QM1SVRCONN) CHLTYPE(SVRCONN) TRPTYPE(TCP)
SSLCIPH(TLS_RSA_WITH_AES_128_CBC_SHA)
```

   Další informace naleznete v tématu Kanál připojení serveru .
9. Definujte a spusťte modul listener kanálu na správci front. Další informace viz DEFINE LISTENER a START LISTENER.
10. Spusťte ukázkový program pomocí následujícího příkazu:

```
AMQSSSLC -m QM1 -c QM1SVRCONN -x localhost
-k "C:\Program Files\IBM\MQ\clientkey" -s TLS_RSA_WITH_AES_128_CBC_SHA
-o http://dummy.OCSP.responder
```

Výsledky

Ukázkový program provádí následující akce:

1. Připojí se k libovolnému určenému správci front nebo k výchozímu správci front s použitím zadaných voleb.
2. Otevře správce front a ve svém názvu zklidní.
3. Zavře správce front.
4. Odpojí se od správce front.

Pokud se ukázkový program spustí úspěšně, zobrazí výstup podobný následujícímu příkladu:

```
Sample AMQSSSLC start
Connecting to queue manager QM1
Using the server connection channel QM1SVRCONN
on connection name localhost.
Using TLS CipherSpec TLS_RSA_WITH_AES_128_CBC_SHA
Using TLS key repository stem C:\Program Files\IBM\MQ\clientkey
Using OCSP responder URL http://dummy.OCSP.responder
Connection established to queue manager QM1

Sample AMQSSSLC end
```

Pokud ukázkový program narazí na problém, zobrazí příslušnou chybovou zprávu, například pokud zadáte neplatnou adresu URL odpovídacího modulu OCSP, obdržíte následující zprávu:

```
MQCONNX ended with reason code 2553
```

Seznam kódů příčiny naleznete v tématu Kód příčiny a kódy příčiny rozhraní API.

Spouštěcí ukázkové programy

Funkce poskytnutá ve spouštěcí ukázce je podmnožinou, která je poskytována v monitoru spouštěčů v programu **runmqtrm** .

Názvy těchto programů viz “Vlastnosti demonstrovány v ukázkových programech na platformách Multiplatforms” na stránce 1036 .

Návrh spouštěcí ukázky

Spouštěcí ukázkový program otevře inicializační frontu s použitím volání MQOPEN s volbou MQOO_INPUT_AS_Q_DEF. Zpráva získává zprávy z inicializační fronty pomocí příkazu MQGET s volbami MQGMO_ACCEPT_TRUNCATED_MSG a MQGMO_WAIT, přičemž určuje neomezený interval čekání. Program vymaže pole *MsgId* a *CorrelId* před každým voláním MQGET, aby získal zprávy v posloupnosti.

Po načtení zprávy z inicializační fronty program otestuje zprávu tak, že zkontroluje velikost zprávy a ujistěte se, že má stejnou velikost jako struktura MQTM. Pokud tento test selže, zobrazí se v programu varování.

V případě platných zpráv spouštěče spouštěcí ukázka kopíruje data z těchto polí: *ApplicId*, *EnvrData*, *Version*a *ApplType*. Poslední dvě z těchto polí jsou numerická, takže program vytvoří náhradu znaků pro použití ve struktuře MQTMC2 pro systémy IBM i, UNIX, Linux, and Windows .

Ukázka spouštěče vydá spouštěcí příkaz do aplikace zadané v poli *ApplicId* zprávy spouštěče a předává strukturu MQTMC2 nebo MQTMC (znaková verze zprávy spouštěcího impulsu).

- ULW V systému UNIX, Linux, and Windows se pole *EnvData* používá jako rozšíření pro řetězec příkazu pro vyvolání.
- IBM i V produktu IBM ise používá jako parametry pro zadání úlohy, například priorita úlohy nebo popis úlohy.

Nakonec program zavře inicializační frontu.

Ukončení spouštění ukázkových programů v systému IBM i

IBM i

Program monitoru spouštěčů může být ukončen volbou sysrequest 2 (ENDRQS) nebo tím, že se replikace získá ze spouštěcí fronty.

Je-li použita vzorová spouštěcí fronta, příkaz je:

```
CHGMQMQ QNAME('SYSTEM.SAMPLE.TRIGGER') MQMNAME GETENBL(*NO)
```

Důležité: Před opětovným spuštěním spouštění v této frontě je třeba zadat následující příkaz:

```
CHGMQMQ QNAME('SYSTEM.SAMPLE.TRIGGER') GETENBL(*YES)
```

Spuštění ukázkových programů spouštění

Toto téma obsahuje informace o spouštění ukázkových programů spouštění.

Spuštění ukázek amqstrg0.c, amqstrg, a amqstrgc

Program má 2 parametry:

1. Název inicializační fronty (nezbytné)
2. Název správce front (volitelný)

Není-li správce front zadán, připojí se k výchozímu správci front. Ukázková inicializační fronta bude definována při spuštění příkazu amqscos0.tst; název této fronty: SYSTEM.SAMPLE.TRIGGERa vy jej můžete použít při spuštění tohoto programu.

Poznámka: Funkce v této ukázce je podmnožinou úplné spouštěcí funkce, která je dodána v programu runmqtrm.

Spuštění ukázky AMQSTRG4

IBM i

Toto je monitor spouštěčů pro prostředí IBM i . Předává jednu úlohu IBM i pro každou aplikaci, která má být spuštěna. To znamená, že existuje další zpracování přidružené ke každé zprávě spouštěče.

AMQSTRG4 (v QCSRC) má dva parametry: název inicializační fronty, která má sloužit, a jméno správce front (volitelné). Příkaz AMQSAMP4 (v QCLSRC) definuje ukázkovou inicializační frontu SYSTEM.SAMPLE.TRIGGER, který můžete použít, když zkoušíte ukázkové programy.

Pomocí ukázkové fronty triggeru je příkaz k vydání:

```
CALL PGM(QMQM/AMQSTRG4) PARM('SYSTEM.SAMPLE.TRIGGER')
```

Případně můžete použít ekvivalent CL STRMQMTRM; podrobnosti viz téma Spuštění monitoru spouštěčů MQ (STRMQMTRM).

Spuštění ukázky AMQSERV4

IBM i

Toto je spouštěcí server pro prostředí IBM i . Pro každou zprávu spouštěče spouští tento server ve své vlastní úloze spouštěcí příkaz ke spuštění zadané aplikace. Spouštěcí server může volat transakce CICS .

Příkaz AMQSERV4 má dva parametry: název inicializační fronty, která má sloužit, a název správce front (volitelné). AMQSAMP4 definuje ukázkovou inicializační frontu SYSTEM.SAMPLE.TRIGGER, který můžete použít, když zkoušíte ukázkové programy.

Pomocí vzorového spouštěcího impulsu, který má příkaz vydat, je:

```
CALL PGM(QMQM/AMQSERV4) PARM('SYSTEM.SAMPLE.TRIGGER')
```

Návrh spouštěcího serveru
Návrh spouštěcího serveru je podobný jako v případě monitoru spouštěčů, s několika výjimkami

Návrh spouštěcího serveru je podobný jako u monitoru spouštěčů, kromě toho, že spouštěcí server:

- Umožňuje použití aplikací MQAT_CICS a aplikací MQAT_OS400 .
- IBM i Volá aplikaci IBM i ve své vlastní úloze (nebo používá příkaz STRCICSUSR ke spuštění aplikací CICS), spíše než aby zadávala úlohu IBM i .
- U aplikací produktu CICS nahradí *EnvData*, například, aby určoval oblast CICS , ze zprávy spouštěče v příkazu STRCICSUSR.
- Otevře inicializační frontu pro sdílený vstup, takže mnoho serverů spouštěčů může být spuštěno ve stejnou dobu.

Poznámka: Programy, které spustil AMQSERV4 , nesmí používat volání MQDISC, protože tento server zastaví spouštěcí server. Pokud programy spouštěné systémem AMQSERV4 používají volání MQCONN, získají kód příčiny MQRC_ALREADY_CONNECTED.

Windows UNIX *Použití ukázek TUXEDO na systémech UNIX a Windows*

Zde najdete informace o ukázkových programech pro operaci Put and Get pro prostředí TUXEDO a pro sestavení prostředí serveru v TUXEDO.

Než začnete

Před spuštěním těchto ukázek je třeba vytvořit prostředí serveru.

Informace o této úloze

Poznámka: V celé této sekci se znak zpětného lomítka (\) používá k rozdělení dlouhých příkazů na více než jeden řádek. Nezadávejte tento znak. Zadejte každý příkaz jako jeden řádek.

Windows UNIX *Vytváření prostředí serveru*
Informace o vytváření prostředí serveru pro produkt IBM MQ pro různé platformy.

Než začnete

Předpokládá se, že máte fungující prostředí TUXEDO.

AIX *Sestavení prostředí serveru pro produkt AIX (32bitový)*
Jak sestavit prostředí serveru pro produkt IBM MQ for AIX (32bitový).

Postup

1. Vytvořte adresář (například APPDIR), ve kterém je založeno prostředí serveru, a proveďte všechny příkazy v tomto adresáři.
2. Exportujte následující proměnné prostředí, kde TUXDIR je kořenový adresář pro TUXEDO, a *MQ_INSTALLATION_PATH* představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ :

```
$ export CFLAGS="-I MQ_INSTALLATION_PATH/inc -I /APPDIR -L MQ_INSTALLATION_PATH/lib"
$ export LDOPTS="-lmqm"
$ export FIELDTBLS= MQ_INSTALLATION_PATH/samp/amqstxvx.flds
$ export VIEWFILES=/APPDIR/amqstxvx.V
$ export LIBPATH=$TUXDIR/lib: MQ_INSTALLATION_PATH/lib:/lib
```

3. Přidejte následující řádek do souboru TUXEDO udataobj/RM:

```
MQSeries_XA_RMI:MQRMIXASwitchDynamic: -lmqmxa -lmqm
```

4. Spusťte příkazy:

```
$ mkfldhdr    MQ_INSTALLATION_PATH/samp/amqstxvx.flds
$ viewc       MQ_INSTALLATION_PATH/samp/amqstxvx.v
$ buildtms    -o MQXA -r MQSeries_XA_RMI
$ buildserver -o MQSERV1 -f MQ_INSTALLATION_PATH/samp/amqstxsx.c \
       -f MQ_INSTALLATION_PATH/lib/libmqm.a \
       -r MQSeries_XA_RMI -s MPUT1:MPUT \
       -s MGET1:MGET \
       -v -bshm
$ buildserver -o MQSERV2 -f MQ_INSTALLATION_PATH/samp/amqstxsx.c \
       -f MQ_INSTALLATION_PATH/lib/libmqm.a \
       -r MQSeries_XA_RMI -s MPUT2:MPUT
       -s MGET2:MGET \
       -v -bshm
$ buildclient -o doputs -f MQ_INSTALLATION_PATH/samp/amqstxpx.c \
       -f MQ_INSTALLATION_PATH/lib/libmqm.a
$ buildclient -o dogets -f MQ_INSTALLATION_PATH/samp/amqstxgx.c \
       -f MQ_INSTALLATION_PATH/lib/libmqm.a
```

5. Upravte soubor ubbstxcx.cfg a v případě potřeby přidejte podrobnosti o názvu počítače, pracovních adresářích a správci front:

```
$ tmloadcf  -y MQ_INSTALLATION_PATH/samp/ubbstxcx.cfg
```

6. Vytvořte TLOGDEVICE:

```
$tmadmin -c
```

Zobrazí se výzva k zadání. Na této obrazovce zadejte:

```
> crdl -z /APPDIR/TLOG1
```

7. Spusťte správce front:

```
$ strmqm
```

8. Spustit Tuxedo

```
$ tmboot -y
```

Jak pokračovat dále

Nyní můžete pomocí programů doputs a dogets vkládat zprávy do fronty a načítat je z fronty.

AIX *Sestavení prostředí serveru pro produkt AIX (64bitový)*
Jak vytvořit prostředí serveru pro produkt IBM MQ for AIX (64bitový).

Postup

1. Vytvořte adresář (například APPDIR), ve kterém je založeno prostředí serveru, a proveďte všechny příkazy v tomto adresáři.
2. Exportujte následující proměnné prostředí, kde TUXDIR představuje kořenový adresář pro TUXEDO a *MQ_INSTALLATION_PATH* představuje adresář vysoké úrovně, ve kterém je IBM MQ installed.:

```
$ export CFLAGS="-I MQ_INSTALLATION_PATH/inc -I /APPDIR -L MQ_INSTALLATION_PATH/lib64"
$ export LDOPTS="-lmqm"
$ export FIELDTBLS= MQ_INSTALLATION_PATH/samp/amqstxvx.flds
$ export VIEWFILES=/APPDIR/amqstxvx.V
$ export LIBPATH=$TUXDIR/lib64: MQ_INSTALLATION_PATH/lib64:/lib64
```

3. Přidejte následující řádek do souboru TUXEDO udataobj/RM:

```
MQSeries_XA_RMI:MQRMIXASwitchDynamic: -lmqmxa64 -lmqm
```

4. Spusťte příkazy:

```
$ mkfldhdr    MQ_INSTALLATION_PATH/samp/amqstxvx.flds
$ viewc       MQ_INSTALLATION_PATH/samp/amqstxvx.v
$ buildtms    -o MQXA -r MQSeries_XA_RMI
$ buildserver -o MQSERV1 -f MQ_INSTALLATION_PATH/samp/amqstxsx.c \
       -f MQ_INSTALLATION_PATH/lib64/libmqm.a \
       -r MQSeries_XA_RMI -s MPUT1:MPUT \
       -s MGET1:MGET \
       -v -bshm
$ buildserver -o MQSERV2 -f MQ_INSTALLATION_PATH/samp/amqstxsx.c \
       -f MQ_INSTALLATION_PATH/lib64/libmqm.a \
       -r MQSeries_XA_RMI -s MPUT2:MPUT
       -s MGET2:MGET \
       -v -bshm
$ buildclient -o doputs -f MQ_INSTALLATION_PATH/samp/amqstxpx.c \
       -f MQ_INSTALLATION_PATH/lib64/libmqm.a
$ buildclient -o dogets -f MQ_INSTALLATION_PATH/samp/amqstxgx.c \
       -f MQ_INSTALLATION_PATH/lib64/libmqm.a
```

5. Upravte soubor ubbstxcx.cfg a v případě potřeby přidejte podrobnosti o názvu počítače, pracovních adresářích a správci front:

```
$ tmloadcf  -y MQ_INSTALLATION_PATH/samp/ubbstxcx.cfg
```

6. Vytvořte TLOGDEVICE:

```
$tmadmin -c
```

Zobrazí se výzva k zadání. Na této obrazovce zadejte:

```
> crdl -z /APPDIR/TLOG1
```

7. Spusťte správce front:

```
$ strmqm
```

8. Spustit Tuxedo

```
$ tmboot -y
```

Jak pokračovat dále

Nyní můžete pomocí programů doputs a dogets vkládat zprávy do fronty a načítat je z fronty.

HP-UX *Sestavení prostředí serveru pro produkt HP-UX (32bitový)*

Jak sestavit prostředí serveru pro produkt IBM MQ for HP-UX (32bitový).

Informace o této úloze

Poznámka: 32bitovou prostředí serveru TUXEDO lze sestavit pouze na platformě Itanium .

Postup

1. Vytvořte adresář (například APPDIR), ve kterém je založeno prostředí serveru, a proveďte všechny příkazy v tomto adresáři.
2. Vyexportujte následující proměnné prostředí, kde TUXDIR je kořenový adresář pro TUXEDO:

```
$ export CFLAGS="-Aa -D_HPUX_SOURCE"
$ export FIELDTBLS= MQ_INSTALLATION_PATH/samp/amqstxvx.flds
$ export VIEWFILES=$APPDIR/amqstxvx.V
$ export TUXCONFIG=$APPDIR/tuxconfig
$ export PATH=$TUXDIR/bin:/usr/bin:/sbin: MQ_INSTALLATION_PATH/bin:$PATH
$ export SHLIB_PATH=$TUXDIR/lib: MQ_INSTALLATION_PATH/lib:/lib
$ export FLDTBLDIR=$APPDIR:$TUXDIR/udataobj
```

3. Přidejte následující řádek do souboru TUXEDO udataobj/RM:

```
MQSeries_XA_RMI:MQRMIXASwitchDynamic: \
MQ_INSTALLATION_PATH/lib/libmqmxa.so MQ_INSTALLATION_PATH/lib/libmqm.so \
/opt/tuxedo/lib/libtux.sl
```

4. Spusťte příkazy:

```
$ mkfldhdr MQ_INSTALLATION_PATH/samp/amqstxvx.flds
$ viewc    MQ_INSTALLATION_PATH/samp/amqstxvx.v
```

Po spuštění příkazů mkfldhdr a viewc se vytvoří soubor záhlaví amqstxvx.h v adresáři aplikace TUXEDO. Zkopírujte tento soubor z adresáře aplikace TUXEDO do adresáře include TUXEDO a pak spusťte následující příkazy.

```
$ buildtms    -o MQXA -r MQSERIES_XA_RMI
$ buildserver -o MQSERV1 -f MQ_INSTALLATION_PATH/samp/amqstxsx.c \
       -f MQ_INSTALLATION_PATH/lib/libmqm.so \
       -r MQSERIES_XA_RMI -s MPUT1:MPUT \
```

```
       -s MGET1:MGET \
       -v -bshm
$ buildserver -o MQSERV2 -f MQ_INSTALLATION_PATH/samp/amqstxsx.c \
       -f MQ_INSTALLATION_PATH/lib/libmqm.so \
       -r MQSERIES_XA_RMI -s MPUT2:MPUT \
       -s MGET2:MGET \
       -v -bshm
$ buildclient -o doputs -f MQ_INSTALLATION_PATH/samp/amqstxpx.c \
       -f MQ_INSTALLATION_PATH/lib/libmqm.so
$ buildclient -o dogets -f MQ_INSTALLATION_PATH/samp/amqstxgx.c \
       -f MQ_INSTALLATION_PATH/lib/libmqm.so
```

5. Upravte ubbstxcx.cfg a podle potřeby přidejte podrobnosti o názvu počítače, pracovních adresářích a správci front:

```
$ tmloadcf  -y MQ_INSTALLATION_PATH/samp/ubbstxcx.cfg
```

6. Vytvořte TLOGDEVICE:

```
$tmadmin -c
```

   Zobrazí se výzva k zadání. Na této obrazovce zadejte:

```
> crdl -z /APPDIR/TLOG1
```

7. Spusťte správce front:

```
$ strmqm
```

8. Spustit TUXEDO:

```
$ tmboot -y
```

Jak pokračovat dále

Nyní můžete pomocí programů doputs a dogets vkládat zprávy do fronty a načítat je z fronty.

Sestavení prostředí serveru pro produkt HP-UX (64bitový)

Jak vytvořit prostředí serveru pro produkt IBM MQ for HP-UX (64bitový).

Informace o této úloze

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Postup

1. Vytvořte adresář (například APPDIR), ve kterém je založeno prostředí serveru, a proveďte všechny příkazy v tomto adresáři.
2. Vyexportujte následující proměnné prostředí, kde TUXDIR je kořenový adresář pro TUXEDO:

```
$ export CFLAGS="-Aa -D_HPUX_SOURCE"
$ export FIELDTBLS= MQ_INSTALLATION_PATH/samp/amqstxvx.flds
$ export VIEWFILES=$APPDIR/amqstxvx.V
$ export TUXCONFIG=$APPDIR/tuxconfig
$ export PATH=$TUXDIR/bin:/usr/bin:/sbin: MQ_INSTALLATION_PATH/bin:$PATH
$ export SHLIB_PATH=$TUXDIR/lib: MQ_INSTALLATION_PATH/lib64:/lib64
$ export FLDTBLDIR=$APPDIR:$TUXDIR/udataobj
```

3. Na platformě HP-UX IA64 (IPF) přidejte níže uvedený řádek do souboru TUXEDO udataobj/RM:

```
MQSeries_XA_RMI:MQRMIXASwitchDynamic: \
MQ_INSTALLATION_PATH/lib64/libmqmxa64.so MQ_INSTALLATION_PATH/lib64/libmqm.so \
/opt/tuxedo/lib/libtux.sl
```

Poznámka: Knihovny IBM MQ dodané na platformě HP-UX IA64 (IPF) mají příponu názvu souboru .so.

4. Spusťte příkazy:

```
$ mkfldhdr MQ_INSTALLATION_PATH/samp/amqstxvx.flds
$ viewc    MQ_INSTALLATION_PATH/samp/amqstxvx.v
```

Po spuštění příkazů mkfldhdr a viewc se vytvoří soubor záhlaví `amqstxvx.h` v adresáři aplikace TUXEDO. Zkopírujte tento soubor z adresáře aplikace TUXEDO do adresáře include TUXEDO a pak spusťte následující příkazy.

```
$ buildtms  -o MQXA -r MQSERIES_XA_RMI
```

Na platformě HP-UX IA64 (IPF):

```
$ buildserver -o MQSERV1 -f MQ_INSTALLATION_PATH/samp/amqstxsx.c \
       -f MQ_INSTALLATION_PATH/lib64/libmqm.so \
       -r MQSERIES_XA_RMI -s MPUT1:MPUT \
       -s MGET1:MGET \
       -v -bshm
$ buildserver -o MQSERV2 -f MQ_INSTALLATION_PATH/samp/amqstxsx.c \
       -f MQ_INSTALLATION_PATH/lib64/libmqm.so \
       -r MQSERIES_XA_RMI -s MPUT2:MPUT \
       -s MGET2:MGET \
       -v -bshm
$ buildclient -o doputs -f MQ_INSTALLATION_PATH/samp/amqstxpx.c \
       -f MQ_INSTALLATION_PATH/lib64/libmqm.so
$ buildclient -o dogets -f MQ_INSTALLATION_PATH/samp/amqstxgx.c \
       -f MQ_INSTALLATION_PATH/lib64/libmqm.so
```

5. Upravte soubor ubbstxcx.cfg a v případě potřeby přidejte podrobnosti o názvu počítače, pracovních adresářích a správci front:

```
$ tmloadcf  -y MQ_INSTALLATION_PATH/samp/ubbstxcx.cfg
```

6. Vytvořte TLOGDEVICE:

```
$tmadmin -c
```

Zobrazí se výzva k zadání. Na této obrazovce zadejte:

```
> crdl -z /APPDIR/TLOG1
```

7. Spusťte správce front:

```
$ strmqm
```

8. Spustit TUXEDO:

```
$ tmboot -y
```

Jak pokračovat dále

Nyní můžete pomocí programů doputs a dogets vkládat zprávy do fronty a načítat je z fronty.

Solaris *Sestavení prostředí serveru pro produkt Solaris (32bitový)*
Jak sestavit prostředí serveru pro produkt IBM MQ for Solaris (32bitový).

Informace o této úloze

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Postup

1. Vytvořte adresář (například APPDIR), ve kterém je založeno prostředí serveru, a proveďte všechny příkazy v tomto adresáři.
2. Vyexportujte následující proměnné prostředí, kde TUXDIR je kořenový adresář pro TUXEDO:

```
$ export CFLAGS="-I /APPDIR"
$ export FIELDTBLS=amqstxvx.flds
$ export VIEWFILES=amqstxvx.V
$ export SHLIB_PATH=$TUXDIR/lib:MQ_INSTALLATION_PATH/lib:/lib
$ export LD_LIBRARY_PATH=$TUXDIR/lib:MQ_INSTALLATION_PATH/lib:/lib
```

3. Přidejte následující do souboru TUXEDO udataobj/RM (RM musí zahrnovat *MQ_INSTALLATION_PATH*/lib/libmqmcs a *MQ_INSTALLATION_PATH*/lib/libmqmzse).

```
MQSeries_XA_RMI:MQRMIXASwitchDynamic: \
MQ_INSTALLATION_PATH/lib/libmqmxa.a MQ_INSTALLATION_PATH/lib/libmqm.so \
/opt/tuxedo/lib/libtux.a MQ_INSTALLATION_PATH/lib/libmqmcs.so \
MQ_INSTALLATION_PATH/lib/libmqmzse.so
```

4. Spusťte příkazy:

```
$ mkfldhdr    amqstxvx.flds
$ viewc       amqstxvx.v
$ buildtms    -o MQXA -r MQSERIES_XA_RMI
$ buildserver -o MQSERV1 -f amqstxsx.c \
       -f MQ_INSTALLATION_PATH/lib/libmqm.so \
       -r MQSERIES_XA_RMI -s MPUT1:MPUT \
       -s MGET1:MGET \
       -v -bshm
       -l -ldl
$ buildserver -o MQSERV2 -f amqstxsx.c \
       -f MQ_INSTALLATION_PATH/lib/libmqm.so \
       -r MQSERIES_XA_RMI -s MPUT2:MPUT \
       -s MGET2:MGET \
       -v -bshm
       -l -ldl
$ buildclient -o doputs -f amqstxpx.c \
       -f MQ_INSTALLATION_PATH/lib/libmqm.so \
       -f MQ_INSTALLATION_PATH/lib/libmqmzse.co \
       -f MQ_INSTALLATION_PATH/lib/libmqmcs.so
$ buildclient -o dogets -f amqstxgx.c \
       -f MQ_INSTALLATION_PATH/lib/libmqm.so
       -f MQ_INSTALLATION_PATH/lib/libmqmzse.co \
       -f MQ_INSTALLATION_PATH/lib/libmqmcs.so
```

5. Upravte ubbstxcx.cfg a podle potřeby přidejte podrobnosti o názvu počítače, pracovních adresářích a správci front:

```
$ tmloadcf  -y ubbstxcx.cfg
```

6. Vytvořte TLOGDEVICE:

```
$tmadmin -c
```

   Zobrazí se výzva k zadání. Na této obrazovce zadejte:

```
> crdl -z /APPDIR/TLOG1
```

7. Spusťte správce front:

```
$ strmqm
```

8. Spustit Tuxedo

```
$ tmboot -y
```

Jak pokračovat dále

Nyní můžete pomocí programů doputs a dogets vkládat zprávy do fronty a načítat je z fronty.

Solaris *Sestavení prostředí serveru pro produkt Solaris (64bitový)*
Jak vytvořit prostředí serveru pro produkt IBM MQ for Solaris (64bitový).

Informace o této úloze

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Postup

1. Vytvořte adresář (například APPDIR), ve kterém je založeno prostředí serveru, a proveďte všechny příkazy v tomto adresáři.
2. Vyexportujte následující proměnné prostředí, kde TUXDIR je kořenový adresář pro TUXEDO:

```
$ export CFLAGS="-I /APPDIR"
$ export FIELDTBLS=amqstxvx.flds
$ export VIEWFILES=amqstxvx.V
$ export SHLIB_PATH=$TUXDIR/lib: MQ_INSTALLATION_PATH/lib:/lib64
$ export LD_LIBRARY_PATH=$TUXDIR/lib64: MQ_INSTALLATION_PATH/lib64:/lib64
```

3. Přidejte následující do souboru TUXEDO udataobj/RM (RM musí zahrnovat *MQ_INSTALLATION_PATH*/lib/libmqmcs a *MQ_INSTALLATION_PATH*/lib/libmqmzse).

```
MQSERIES_XA_RMI:MQRMIXASwitchDynamic: \
MQ_INSTALLATION_PATH/lib64/libmqmxa64.a MQ_INSTALLATION_PATH/lib64/libmqm.so \
/opt/tuxedo/lib64/libtux.a MQ_INSTALLATION_PATH/lib64/libmqmcs.so \
MQ_INSTALLATION_PATH/lib64/libmqmzse.so
```

4. Spusťte příkazy:

```
$ mkfldhdr    amqstxvx.flds
$ viewc       amqstxvx.v
$ buildtms    -o MQXA -r MQSERIES_XA_RMI
$ buildserver -o MQSERV1 -f amqstxsx.c \
       -f MQ_INSTALLATION_PATH/lib64/libmqm.so \
       -r MQSERIES_XA_RMI -s MPUT1:MPUT \
       -s MGET1:MGET \
       -v -bshm
       -l -ldl
$ buildserver -o MQSERV2 -f amqstxsx.c \
       -f MQ_INSTALLATION_PATH/lib64/libmqm.so \
       -r MQSERIES_XA_RMI -s MPUT2:MPUT \
       -s MGET2:MGET \
       -v -bshm
       -l -ldl
$ buildclient -o doputs -f amqstxpx.c \
       -f MQ_INSTALLATION_PATH/lib64/libmqm.so \
       -f MQ_INSTALLATION_PATH/lib64/libmqmzse.co \
```

```
       -f MQ_INSTALLATION_PATH/lib64/libmqmcs.so
$ buildclient -o dogets -f amqstxgx.c \
       -f MQ_INSTALLATION_PATH/lib64/libmqm.so
       -f MQ_INSTALLATION_PATH/lib64/libmqmzse.co \
       -f MQ_INSTALLATION_PATH/lib64/libmqmcs.so
```

5. Upravte ubbstxcx.cfg a podle potřeby přidejte podrobnosti o názvu počítače, pracovních adresářích a správci front:

```
$ tmloadcf  -y ubbstxcx.cfg
```

6. Vytvořte TLOGDEVICE:

```
$tmadmin -c
```

   Zobrazí se výzva k zadání. Na této obrazovce zadejte:

```
> crdl -z /APPDIR/TLOG1
```

7. Spusťte správce front:

```
$ strmqm
```

8. Spustit Tuxedo

```
$ tmboot -y
```

Jak pokračovat dále

Nyní můžete pomocí programů doputs a dogets vkládat zprávy do fronty a načítat je z fronty.

Windows *Sestavení prostředí serveru pro produkt Windows (32bitový)*

Sestavuje se prostředí serveru pro produkt IBM MQ for Windows (32bitový).

Informace o této úloze

Poznámka: Změňte pole označená jako *VARIABLES* v následujících adresářích na cesty k adresářům:

Tabulka 153. Pole pro změnu cest k adresářům

Pole	**Cesta k adresáři**
MQMDIR	Cesta k adresáři uvedená při instalaci produktu IBM MQ , například `g:\Program Files\IBM\MQ`.
TUXDIR	Cesta k adresáři uvedená při instalaci TUXEDO, například `f:\tuxedo`.
ADRESÁŘ	Cesta k adresáři, která má být použita pro ukázkovou aplikaci, například `f:\tuxedo\apps\mqapp`.

```
*RESOURCES
IPCKEY       99999
UID          0
GID          0
MAXACCESSERS 20
MAXSERVERS   20
MAXSERVICES  50
MASTER       SITE1
MODEL        SHM
LDBAL        N

*MACHINES
MachineName LMID=SITE1
            TUXDIR="f:\tuxedo"
            APPDIR="f:\tuxedo\apps\mqapp;g:\Program Files\IBM\WebSphere MQ\bin"
            ENVFILE="f:\tuxedo\apps\mqapp\amqstxen.env"
            TUXCONFIG="f:\tuxedo\apps\mqapp\tuxconfig"
            ULOGPFX="f:\tuxedo\apps\mqapp\ULOG"
            TLOGDEVICE="f:\tuxedo\apps\mqapp\TLOG"
            TLOGNAME=TLOG
            TYPE="i386NT"
            UID=0
            GID=0

*GROUPS
GROUP1
        LMID=SITE1 GRPNO=1
        TMSNAME=MQXA
        OPENINFO="MQSERIES_XA_RMI:MYQUEUEMANAGER"

*SERVERS
DEFAULT: CLOPT="-A -- -m MYQUEUEMANAGER"

MQSERV1   SRVGRP=GROUP1 SRVID=1
MQSERV2   SRVGRP=GROUP1 SRVID=2

*SERVICES
MPUT1
MGET1
MPUT2
MGET2
```

Obrázek 146. Příklad souboru ubbstxcn.cfg pro IBM MQ for Windows

Poznámka: Změňte název počítače *MachineName*a cesty k adresáři tak, aby odpovídaly vaší instalaci. Změňte také název správce front *MYQUEUEMANAGER* na název správce front, ke kterému se chcete připojit.

Ukázkový soubor ubbconfig pro produkt IBM MQ for Windows je uveden v seznamu Obrázek 146 na stránce 1106. Dodává se jako ubbstxcn.cfg v adresáři ukázek produktu IBM MQ .

Ukázkový soubor makefile (viz Obrázek 147 na stránce 1107) dodaný pro IBM MQ for Windows se nazývá ubbstxmn.maka je držen v adresáři ukázek IBM MQ .

```
TUXDIR  = f:\tuxedo
MQMDIR  = g:\Program Files\IBM\WebSphere MQ
APPDIR  = f:\tuxedo\apps\mqapp
MQMLIB  = $(MQMDIR)\tools\lib
MQMINC  = $(MQMDIR)\tools\c\include
MQMSAMP = $(MQMDIR)\tools\c\samples
INC = -f "-I$(MQMINC) -I$(APPDIR)"
DBG = -f "/Zi"

amqstx.exe:
 $(TUXDIR)\bin\mkfldhdr    -d$(APPDIR) $(MQMSAMP)\amqstxvx.fld
 $(TUXDIR)\bin\viewc       -d$(APPDIR) $(MQMSAMP)\amqstxvx.v
 $(TUXDIR)\bin\buildtms    -o MQXA -r MQSERIES_XA_RMI
 $(TUXDIR)\bin\buildserver -o MQSERV1 -f $(MQMSAMP)\amqstxsx.c \
                           -f $(MQMLIB)\mqm.lib -v $(INC) $(DBG) \
                           -r MQSERIES_XA_RMI \
                           -s MPUT1:MPUT -s MGET1:MGET
 $(TUXDIR)\bin\buildserver -o MQSERV2 -f $(MQMSAMP)\amqstxsx.c \
                           -f $(MQMLIB)\mqm.lib -v $(INC) $(DBG) \
                           -r MQSERIES_XA_RMI \
                           -s MPUT2:MPUT -s MGET2:MGET
 $(TUXDIR)\bin\buildclient -o doputs -f $(MQMSAMP)\amqstxpx.c \
                           -f $(MQMLIB)\mqm.lib -v $(INC) $(DBG)
$(TUXDIR)\bin\buildclient  -o dogets -f $(MQMSAMP)\amqstxgx.c \
                           -f $(MQMLIB)\mqm.lib $(INC) -v $(DBG)
 $(TUXDIR)\bin\tmloadcf    -y $(APPDIR)\ubbstxcn.cfg
```

Obrázek 147. Ukázkový soubor makefile TUXEDO pro IBM MQ for Windows

Chcete-li sestavit prostředí serveru a ukázky, postupujte takto.

Postup

1. Vytvořte adresář aplikace, do kterého se má sestavit ukázková aplikace, například:

   ```
   f:\tuxedo\apps\mqapp
   ```

2. Zkopírujte následující ukázkové soubory z ukázkového adresáře IBM MQ do adresáře aplikace:
   - `amqstxmn.mak`
   - `amqstxen.env`
   - `ubbstxcn.cfg`
3. Upravte každý z těchto souborů a nastavte názvy adresářů a cesty k adresářům použité při instalaci.
4. Upravte `ubbstxcn.cfg` (viz Obrázek 146 na stránce 1106), abyste přidali podrobnosti o názvu počítače a správci front, ke kterému se chcete připojit.
5. Přidejte následující řádek do souboru TUXEDO *TUXDIR*`udataobj\rm`:

   ```
   MQSERIES_XA_RMI;MQRMIXASwitchDynamic;MQMDIR\tools\lib\mqmxa.lib MQMDIR\tools\lib\mqm.lib
   ```

   Nový záznam musí být jeden řádek v souboru.
6. Nastavte následující proměnné prostředí:

   ```
   TUXDIR=TUXDIR
   TUXCONFIG=APPDIR\tuxconfig
   FIELDTBLS=MQMDIR\tools\c\samples\amqstxvx.fld
   LANG=C
   ```

7. Vytvořte zařízení TLOG pro TUXEDO.

   Chcete-li to provést, vyvolejte příkaz `tmadmin -c` a zadejte následující příkaz:

   ```
   crdl -z APPDIR\TLOG
   ```

8. Nastavte aktuální adresář na *APPDIR*a vyvolejte ukázkový soubor Makefile `amqstxmn.mak` jako soubor Makefile externího projektu. Například, s Microsoft Visual C++, zadejte následující příkaz:

   ```
   msvc amqstxmn.mak
   ```

   Vyberte volbu **sestavení** , chcete-li sestavit všechny ukázkové programy.

Windows *Sestavení prostředí serveru pro produkt Windows (64bitový)*
Jak vytvořit prostředí serveru pro produkt IBM MQ for Windows (64bitový).

Informace o této úloze

Poznámka: Změňte pole označená jako *VARIABLES* v následujících adresářích na cesty k adresářům:

Tabulka 154. Pole pro změnu cest k adresářům

Pole	**Cesta k adresáři**
MQMDIR	Cesta k adresáři uvedená při instalaci produktu IBM MQ , například `g:\Program Files\IBM\MQ`.
TUXDIR	Cesta k adresáři uvedená při instalaci TUXEDO, například `f:\tuxedo`.
ADRESÁŘ	Cesta k adresáři, která má být použita pro ukázkovou aplikaci, například `f:\tuxedo\apps\mqapp`.

```
*RESOURCES
IPCKEY       99999
UID          0
GID          0
MAXACCESSERS 20
MAXSERVERS   20
MAXSERVICES  50
MASTER       SITE1
MODEL        SHM
LDBAL        N

*MACHINES
MachineName LMID=SITE1
            TUXDIR="f:\tuxedo"
            APPDIR="f:\tuxedo\apps\mqapp;g:\Programï¿½Files\IBM\WebSphere MQ\bin"
            ENVFILE="f:\tuxedo\apps\mqapp\amqstxen.env"
            TUXCONFIG="f:\tuxedo\apps\mqapp\tuxconfig"
            ULOGPFX="f:\tuxedo\apps\mqapp\ULOG"
            TLOGDEVICE="f:\tuxedo\apps\mqapp\TLOG"
            TLOGNAME=TLOG
            TYPE="i386NT"
            UID=0
            GID=0

*GROUPS
GROUP1
        LMID=SITE1 GRPNO=1
        TMSNAME=MQXA
        OPENINFO="MQSERIES_XA_RMI:MYQUEUEMANAGER"

*SERVERS
DEFAULT: CLOPT="-A -- -m MYQUEUEMANAGER"

MQSERV1   SRVGRP=GROUP1 SRVID=1
MQSERV2   SRVGRP=GROUP1 SRVID=2

*SERVICES
MPUT1
MGET1
MPUT2
MGET2
```

Obrázek 148. Příklad souboru ubbstxcn.cfg pro IBM MQ for Windows

Poznámka: Změňte název počítače *MachineName*a cesty k adresáři tak, aby odpovídaly vaší instalaci. Změňte také název správce front *MYQUEUEMANAGER* na název správce front, ke kterému se chcete připojit.

Ukázkový soubor ubbconfig pro produkt IBM MQ for Windows je uveden v seznamu Obrázek 148 na stránce 1109. Dodává se jako `ubbstxcn.cfg` v adresáři ukázek produktu IBM MQ .

Ukázkový soubor Makefile (viz Obrázek 149 na stránce 1110) Dodáno pro IBM MQ for Windows se nazývá `ubbstxmn.mak`a je zadrženo v adresáři ukázek IBM MQ .

```
TUXDIR  = f:\tuxedo
MQMDIR  = g:\Program Files\IBM\WebSphere MQ
APPDIR  = f:\tuxedo\apps\mqapp
MQMLIB  = $(MQMDIR)\tools\lib64
MQMINC  = $(MQMDIR)\tools\c\include
MQMSAMP = $(MQMDIR)\tools\c\samples
INC = -f "-I$(MQMINC) -I$(APPDIR)"
DBG = -f "/Zi"

amqstx.exe:
 $(TUXDIR)\bin\mkfldhdr    -d$(APPDIR) $(MQMSAMP)\amqstxvx.fld
 $(TUXDIR)\bin\viewc       -d$(APPDIR) $(MQMSAMP)\amqstxvx.v
 $(TUXDIR)\bin\buildtms    -o MQXA -r MQSERIES_XA_RMI
 $(TUXDIR)\bin\buildserver -o MQSERV1 -f $(MQMSAMP)\amqstxsx.c \
                           -f $(MQMLIB)\mqm.lib -v $(INC) $(DBG) \
                           -r MQSERIES_XA_RMI \
                           -s MPUT1:MPUT -s MGET1:MGET
 $(TUXDIR)\bin\buildserver -o MQSERV2 -f $(MQMSAMP)\amqstxsx.c \
                           -f $(MQMLIB)\mqm.lib -v $(INC) $(DBG) \
                           -r MQSERIES_XA_RMI \
                           -s MPUT2:MPUT -s MGET2:MGET
 $(TUXDIR)\bin\buildclient -o doputs -f $(MQMSAMP)\amqstxpx.c \
                           -f $(MQMLIB)\mqm.lib -v $(INC) $(DBG)
 $(TUXDIR)\bin\buildclient -o dogets -f $(MQMSAMP)\amqstxgx.c \
                           -f $(MQMLIB)\mqm.lib $(INC) -v $(DBG)
 $(TUXDIR)\bin\tmloadcf    -y $(APPDIR)\ubbstxcn.cfg
```

Obrázek 149. Ukázkový soubor makefile TUXEDO pro IBM MQ for Windows

Chcete-li sestavit prostředí serveru a ukázky, postupujte takto.

Postup

1. Vytvořte adresář aplikace, do kterého se má sestavit ukázková aplikace, například:

   ```
   f:\tuxedo\apps\mqapp
   ```

2. Zkopírujte následující ukázkové soubory z ukázkového adresáře IBM MQ do adresáře aplikace:
   - `amqstxmn.mak`
   - `amqstxen.env`
   - `ubbstxcn.cfg`
3. Upravte každý z těchto souborů a nastavte názvy adresářů a cesty k adresářům použité při instalaci.
4. Upravit položku `ubbstxcn.cfg` (viz Obrázek 148 na stránce 1109) Chcete-li přidat podrobnosti o názvu počítače a správci front, ke kterému se chcete připojit.
5. Přidejte následující řádek do souboru TUXEDO *TUXDIR*`udataobj\rm`

   ```
   MQSERIES_XA_RMI;MQRMIXASwitchDynamic;MQMDIR\tools\lib64\mqmxa64.lib
   MQMDIR\tools\lib64\mqm.lib
   ```

   Nový záznam musí být jeden řádek v souboru.
6. Nastavte následující proměnné prostředí:

   ```
   TUXDIR=TUXDIR
   TUXCONFIG=APPDIR\tuxconfig
   FIELDTBLS=MQMDIR\tools\c\samples\amqstxvx.fld
   LANG=C
   ```

7. Vytvořte zařízení TLOG pro TUXEDO. Chcete-li to provést, vyvolejte příkaz `tmadmin -c` a zadejte příkaz:

   ```
   crdl -z APPDIR\TLOG
   ```

8. Nastavte aktuální adresář na *APPDIR*a vyvolejte ukázkový soubor Makefile `amqstxmn.mak` jako soubor Makefile externího projektu. Například, s Microsoft Visual C++, zadejte následující příkaz:

```
msvc amqstxmn.mak
```

   Vyberte volbu **sestavení** , chcete-li sestavit všechny ukázkové programy.

Windows UNIX *Ukázkový serverový program pro TUXEDO*

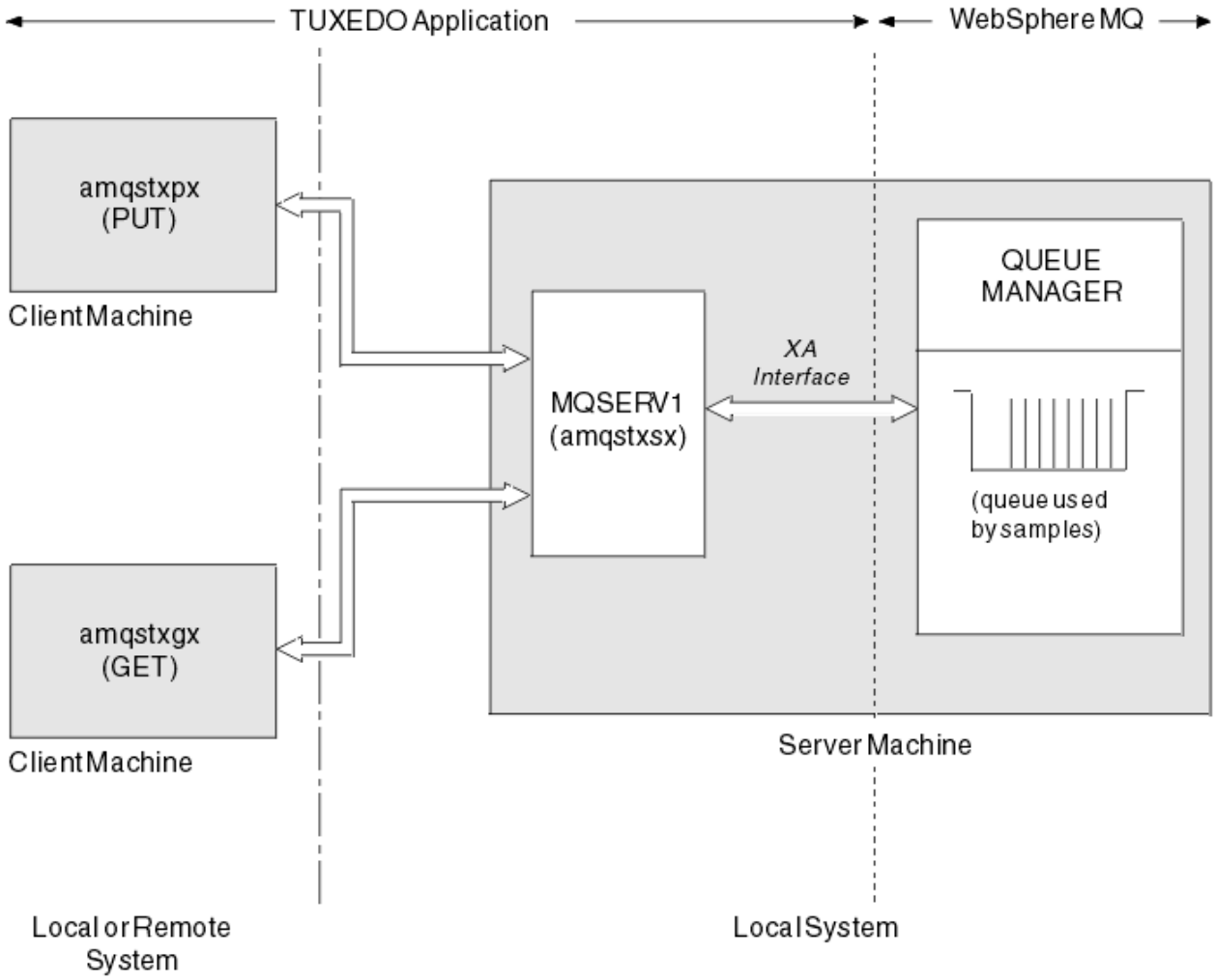
Ukázkový program serveru (amqstxsx) je navržen pro práci s ukázkovými programy Put (amqstxpx.c) a Get (amqstxgx.c). Ukázkový program serveru se spustí automaticky, když se spustí TUXEDO.

Poznámka: Před spuštěním programu TUXEDO je třeba spustit správce front.

Ukázkový server poskytuje dvě služby TUXEDO, MPUT1 a MGET1:

- Služba MPUT1 je řízena ukázkou PUT a používá příkaz MQPUT1 v synchronizačním bodu k vložení zprávy do jednotky práce, kterou řídí TUXEDO. Vezme parametry QName a Text zprávy, které jsou dodány ukázkou PUT.
- Služba MGET1 se otevře a zavře frontu pokaždé, když získá zprávu. Vezme parametry QName a Text zprávy, které jsou dodány ukázkou GET.

Všechny chybové zprávy, kódy příčiny a stavové zprávy jsou zapsány do souboru protokolu TUXEDO.



Obrázek 150. Jak se vzorky TUXEDO vzájemně spolupracují

Windows UNIX *Vložit vzorový program pro TUXEDO*

Tato ukázka vám umožňuje vložit zprávu do fronty vícekrát, v dávkách, demonstrovat pomocí protokolu TUXEDO jako správce prostředků syncpointing.

Ukázkový program serveru `amqstxsx` musí být spuštěn, aby byl ukázkový ukázkový soubor úspěšný; ukázkový program serveru se připojí ke správci front a použije rozhraní XA. Chcete-li spustit ukázku, zadejte:

- `doputs -n queuename -b batchsize -c trancount -t message`

Příklad:

- `doputs -n myqueue -b 5 -c 6 -t "Hello World"`

To vloží 30 zpráv do fronty s názvem `myqueue`, v šesti dávkách, každý s pěti zprávami v něm. Pokud dojde k problémům, zazálohuje dávku zpráv, jinak je potvrdí.

Jakékoli chybové zprávy jsou zapsány do souboru protokolu TUXEDO a na stderr. Jakékoli kódy příčiny jsou zapisovány do stderr.

Windows UNIX *Získat ukázku pro TUXEDO*

Tato ukázka vám umožňuje získat zprávy z fronty v dávkách.

Ukázkový program serveru `amqstxsx` musí být spuštěn, aby byla ukázka Get úspěšná; ukázkový program serveru se připojí ke správci front a použije rozhraní XA. Chcete-li spustit ukázku, zadejte následující příkaz:

- `dogets -n queuename -b batchsize -c trancount`

Příklad:

- `dogets -n myqueue -b 6 -c 4`

To zabere 24 zpráv z fronty s názvem `myqueue`, v šesti dávkách, každá se čtyřmi zprávami v ní. Spustíte-li tento příkaz po vložení příkladu, který vloží 30 zpráv v systému `myqueue`, máte na serveru `myqueue`pouze šest zpráv. Počet dávek a velikost dávky se mohou lišit mezi vložením zpráv a získáním těchto dávek.

Jakékoli chybové zprávy jsou zapsány do souboru protokolu TUXEDO a na stderr. Jakékoli kódy příčiny jsou zapisovány do stderr.

Windows *Použití uživatelské procedury zabezpečení SSPI v systému Windows*

Toto téma popisuje, jak používat programy kanálů SSPI v systémech Windows . Dodaný kód ukončení je ve dvou formátech: objekt a zdroj.

Kód objektu

Soubor s kódem objektu se nazývá `amqrspin.dll`. Pro klienta i server je instalován jako standardní část IBM MQ for Windows ve složce *MQ_INSTALLATION_PATH*`\exits\`*INSTALLATION_NAME* . Například `C:\Program Files\IBM\MQ\exits\installation2`. Je načten jako standardní uživatelský vstup. Předaný konec kanálu zabezpečení můžete spustit a použít ověřovací služby ve své definici kanálu.

Chcete-li to provést, zadejte jednu z následujících možností:

```
SCYEXIT('amqrspin(SCY_KERBEROS)')
SCYEXIT('amqrspin(SCY_NTLM)')
```

Chcete-li poskytnout podporu pro omezený kanál, zadejte na kanálu SVRCONN následující:

```
SCYDATA('remote_principal_name')
```

kde *název_vzdáleného_činitele* je ve tvaru *DOMAIN\uživatel*. Zabezpečený kanál je vytvořen pouze v případě, že se název vzdáleného činitele shoduje s názvem *název_vzdáleného_činitele*.

Chcete-li mezi systémy, které jsou provozovány v doméně zabezpečení Kerberos , používat dodané uživatelské programy kanálu, vytvořte **servicePrincipalName** pro správce front.

Zdrojový kód

Soubor zdrojového kódu ukončení se nazývá `amqsspin.c`. Nachází se v `C:\Program Files\IBM\MQ\Tools\c\Samples`.

Změníte-li zdrojový kód, musíte upravený zdroj znovu zkompilovat.

Kompilujete a propojíte jej stejným způsobem jako ostatní uživatelské procedury kanálu pro příslušnou platformu, kromě toho, že záhlaví SSPI musí být přístupná v době kompilace a knihovny zabezpečení SSPI spolu s doporučenými přidruženými knihovnami je třeba k nim přistupovat v době propojení.

Než provedete následující příkaz, ujistěte se, že `cl.exe`a knihovna Visual C++ a složka `include` jsou dostupné ve vaší cestě. Příklad:

```
cl /VERBOSE /LD /MT /Ipath_to_Microsoft_platform_SDK\include
/Ipath_to_IBM_MQ\tools\c\include amqsspin.c /DSECURITY_WIN32
-link /DLL /EXPORT:SCY_KERBEROS /EXPORT:SCY_NTLM STACK:8192
```

Poznámka: Zdrojový kód neobsahuje žádné ustanovení pro trasování nebo ošetření chyb. Pokud upravíte a použijete zdrojový kód, přidejte své vlastní trasovací a manipulační rutiny.

Spuštění ukázek pomocí vzdálených front

Vzdálené řazení do front lze demonstrovat spuštěním ukázek v připojených správcích front.

Program amqscos0.tst poskytuje lokální definici vzdálené fronty (SYSTEM.SAMPLE.REMOTE), který používá vzdáleného správce front s názvem OTHER. Chcete-li použít tuto definici ukázky, změňte hodnotu OTHER na název druhého správce front, který chcete použít. Musíte také nastavit kanál zpráv mezi dvěma správci front; chcete-li získat informace o tom, jak to provést, prostudujte si téma Definování kanálů.

Vzorové programy požadavku umístí své vlastní lokální názvy správce front do pole *ReplyToQMgr* zpráv, které odesílají. Funkce Inquire and Set slouží k odeslání zpráv odpovědi do fronty a správce front zpráv pojmenované v polích *ReplyToQ* a *ReplyToQMgr* v rámci zpráv požadavků, které zpracovávají.

Ukázkový program pro monitorování front klastru (AMQSCLM)

Tato ukázka používá vestavěné funkce vyrovnávání pracovní zátěže klastru IBM MQ pro přímé zprávy s instancemi front, které mají připojené aplikace. Tento automatický směr zabraňuje sestavení zpráv na instanci fronty klastru, ke které není připojena žádná spotřeba aplikace.

Přehled

Pro stejnou frontu v různých správcích front můžete nastavit klastr, který má více než jednu definici téže fronty. Tato konfigurace poskytuje výhodu zvýšené dostupnosti a vyrovnávání pracovní zátěže. K dynamické úpravě distribuce zpráv v rámci klastru na základě stavu připojených aplikací však není v produktu IBM MQ vestavěna žádná schopnost. Z tohoto důvodu musí být aplikace spotřebitele vždy připojena ke každé instanci fronty, aby bylo zajištěno, že zprávy budou zpracovány.

Ukázkový program monitorování fronty klastru monitoruje stav připojených aplikací. Program dynamicky upravuje konfiguraci vyrovnávání pracovní zátěže pro přímé zprávy na instance klastrované fronty s připojenými aplikacemi spotřebovávající spotřebu. V určitých situacích lze tento program použít k uvolnění nutnosti připojení aplikace tak, aby vždy byla připojena ke každé instanci fronty. Také znovu odešle zprávy, které budou zařazeny do fronty na instanci fronty bez připojených přijímajících aplikací. Opětovné odeslání zpráv umožňuje směrovat zprávy do oblasti spotřebovávající aplikaci, která je dočasně ukončena.

Program je navržen tak, aby jej bylo možné použít tam, kde náročné aplikace jsou dlouhodobě spuštěné aplikace, spíše než často připojovaná a odpojující aplikace.

Ukázkový program pro monitorování fronty klastru je kompilovaným spustitelným programem v ukázkovém souboru C `amqsclma.c`.

Další informace o klastrech a pracovní zátěži naleznete v tématu Použití klastrů pro správu pracovní zátěže .

AMQSCLM: Návrh a plánování pro použití ukázky
Informace o tom, jak pracuje ukázkový program monitorování fronty klastru, je třeba zvážit při nastavení systému pro spuštění ukázkového programu a o úpravách, které lze provést v ukázkovém zdrojovém kódu.

Návrh

Ukázkový program monitorování fronty klastru monitoruje lokální klastrované fronty, které mají připojené aplikace. Program monitoruje fronty určené uživatelem. Název fronty může být specifický, například APP.TEST01, nebo generický. Generické názvy musí být ve formátu, který je v souladu s PCF (Programmable Command Format). Příklady generických názvů jsou APP.TEST*, nebo APP*.

Každý správce front v klastru, který vlastní instanci lokální fronty, která má být monitorována, vyžaduje k připojení instanci ukázkového programu monitorování fronty klastru.

Dynamické směrování zpráv

Ukázkový program monitorování front klastru používá hodnotu fronty **IPPROCS** (otevřeno pro výpočet vstupních procesů) fronty k určení, zda má tato fronta nějaké odběratele. Hodnota větší než 0 označuje, že fronta má připojenu alespoň jednu odebírající aplikaci. Takové fronty jsou aktivní. Hodnota 0 znamená, že fronta nemá žádné připojené náročné programy. Takové fronty jsou neaktivní.

Pro klastrované fronty s více instancemi v klastru produkt IBM MQ používá vlastnost priority zátěže klastru **CLWLPRTY** každé instance fronty k určení instancí, do kterých mají být odesílány zprávy. Příkaz IBM MQ odesílá zprávy do dostupných instancí fronty s nejvyšší hodnotou **CLWLPRTY** .

Ukázkový program pro monitorování fronty klastru aktivuje frontu klastru nastavením lokální hodnoty **CLWLPRTY** na hodnotu 1. Program deaktivuje frontu klastru nastavením jeho hodnoty **CLWLPRTY** na hodnotu 0.

Technologie klastrování produktu IBM MQ šíří aktualizovanou vlastnost produktu **CLWLPRTY** klastrované fronty ke všem relevantním správcům front v klastru. Například

- Správce front s připojenou aplikací, která vkládá zprávy do fronty.
- Správce front, který vlastní lokální frontu se stejným názvem ve stejném klastru.

Šíření se provádí pomocí správců front úplného úložiště klastru. Nové zprávy pro frontu klastru jsou směrovány na instance s nejvyšší hodnotou **CLWLPRTY** v rámci klastru.

Přenos zprávy ve frontě

Dynamická úprava hodnoty **CLWLPRTY** ovlivňuje směrování nových zpráv. Tato dynamická úprava neovlivňuje zprávy, které jsou již ve frontě na instanci fronty bez připojených spotřebitelů, nebo zprávy, které byly přes mechanizmus vyrovnávání pracovní zátěže před šířením upravené hodnoty **CLWLPRTY** , byly propagovány v rámci klastru. V důsledku toho zůstanou zprávy v žádné neaktivní frontě a nebudou zpracovány aplikací spotřebovávající spotřebu. K vyřešení tohoto problému je ukázkový program pro monitorování fronty klastru schopen získat zprávy z lokální fronty bez spotřebitelů a odeslat tyto zprávy vzdáleným instancím stejné fronty, kde jsou spotřebitelé připojeni.

Ukázkový program monitorování fronty klastru přenáší zprávy z neaktivní lokální fronty do jedné nebo více aktivních vzdálených front tím, že získává zprávy (pomocí produktu **MQGET**) a vložení zpráv (pomocí **MQPUT**) do stejné klastrované fronty. Tento přenos způsobí, že správa pracovní zátěže klastru IBM MQ vybere jinou cílovou instanci založenou na vyšší hodnotě CLWLPRTY , než je hodnota instance lokální fronty. Perzistence a kontext zprávy jsou během přenosu zprávy zachovány. Pořadí zpráv a všechny volby vazby nejsou zachovány.

Naplánování

Ukázkový program monitorování fronty klastru upraví konfiguraci klastru, když dojde ke změně v konektivitě aplikací odběratele. Změny jsou přeneseny ze správců front, ve kterých je ukázkový program pro monitorování fronty klastru, do úplných správců front úložiště v klastru. Správci front úplného úložiště zpracují aktualizace konfigurace a znovu je odešlou do všech příslušných správců front v klastru. Relevantní správci front zahrnují tyto správce front, kteří vlastní klastrované fronty se stejným názvem (kde je spuštěna instance ukázkového programu pro monitorování fronty klastru) a správce front, v němž aplikace otevřela frontu klastru k vložení zpráv do fronty za posledních 30 dní.

Změny jsou asynchronně zpracovány v celém klastru. Proto mohou různí správci front v klastru po každé změně mít různé pohledy na konfiguraci po určitou dobu.

Ukázkový program pro monitorování fronty klastru je vhodný pouze pro systémy, kde se spotřebovávají aplikace zřídka připojují nebo odpojují; například dlouho běžící náročné aplikace. Když se používá k monitorování systémů, kde jsou náročné aplikace připojeny pouze pro krátká období, může latence při distribuci aktualizací konfigurace vést k tomu, že správci front v klastru mají nesprávný pohled na fronty, kde jsou spotřebitelé připojeni. Tato latence může vést k nesprávně přesměrovaným zprávám.

Při monitorování mnoha front může relativně nízká míra změn u připojených spotřebitelů ve všech frontách zvýšit provoz konfigurace klastru v rámci klastru. Zvýšený provoz konfigurace klastru může mít za následek nadměrné zatížení na jednom nebo více následujících správcích front.

- Správci front, ve kterém je spuštěn ukázkový program pro monitorování fronty klastru
- Správci front úplného úložiště
- Správce front s připojenou aplikací, která vkládá zprávy do fronty.
- Správce front, který vlastní lokální frontu se stejným názvem ve stejném klastru.

Musí být posouzeno použití procesoru na správci front úplného úložiště. Další využití procesoru je viditelné jako provoz zpráv v úplné frontě úložišť SYSTEM.CLUSTER.COMMAND.QUEUE. Pokud se zprávy v této frontě sestavují, znamená to, že správci front úplného úložiště nejsou schopni udržet krok s rychlostí změny konfigurace klastru v systému.

Při monitorování mnoha front pomocí ukázkového programu pro monitorování fronty klastru je k dispozici množství práce provedené ukázkovým programem a správcem front. Tato práce se provádí i v případě, kdy nejsou k dispozici žádné změny u připojených spotřebitelů. Argument **-i** lze upravit tak, aby se snížilo využití procesoru vzorového programu na lokálním systému, a to snížením frekvence monitorovacího cyklu.

Ukázkový program monitorování fronty klastru pomáhá detekovat nadměrné aktivity a uvádí průměrnou dobu zpracování na interval zjišťování, uplynulý čas zpracování a počet změn konfigurace. Sestavy se doručí v informační zprávě, **CLM0045I**, každých 30 minut nebo každých 600 dotazových intervalů, podle toho, co nastane dříve.

Požadavky na použití monitorování fronty klastru

Ukázkový program monitorování fronty klastru má požadavky a omezení. Vzorový zdrojový kód poskytnutý pro změnu některých těchto omezení můžete upravit tak, jak jej lze použít. Příklady uvedené v podrobných úpravách této sekce, které lze provést.

- Ukázkový program pro monitorování fronty klastru je navržen tak, aby jej bylo možné použít k monitorování front, ve kterých jsou buď připojeny aplikace, které jsou připojeny, nebo nejsou připojeny. Pokud systém spotřebovává aplikace, které jsou často připojovaných a odpojování, může ukázkový program generovat nadměrnou aktivitu konfigurace klastru v celém klastru. To může mít vliv na výkon správců front v klastru.
- Ukázkový program monitorování fronty klastru závisí na základní systémové a klastrové technologii systému IBM MQ . Počet monitorovaných front, četnost monitorování a četnost změn stavu každé fronty ovlivní zatížení na celkovém systému. Tyto faktory je třeba vzít v úvahu při výběru front, které mají být monitorovány, a intervalu výzev k monitorování.

- Instance ukázkového programu pro monitorování fronty klastru musí být připojena ke každému správci front v klastru, který vlastní instanci fronty, která má být monitorována. Není nutné připojit ukázkový program ke správcům front v klastru, který nevlastní fronty.
- Ukázkový program monitorování fronty klastru musí být spuštěn s vhodnou autorizací pro přístup ke všem vyžadovaným prostředkům produktu IBM MQ . Například
  - Správce front, k němuž má být připojen
  - SYSTEM.ADMIN.COMMAND.QUEUE
  - Všechny fronty, které mají být monitorovány, když se provádí přenos zpráv
- Příkazový server musí být spuštěn pro každého správce front se připojeným ukázkovým programem pro monitorování fronty klastru.
- Každá instance ukázkového programu pro monitorování front klastru vyžaduje výlučné použití lokální fronty (neklastrované) ve správci front, k němuž je připojen. Tato lokální fronta se používá k řízení ukázkového programu a přijímá zprávy odpovědí z inquires vytvořených na příkazový server správce front.
- Všechny fronty, které mají být monitorovány jedinou instancí ukázkového programu monitorování fronty klastru, musí být ve stejném klastru. Má-li správce front fronty ve více klastrech, které vyžadují monitorování, je zapotřebí více instancí ukázkového programu. Každá instance potřebuje lokální frontu pro řídicí a odpovědní zprávy.
- Všechny fronty, které mají být monitorovány, musí být v jednom klastru. Fronty konfigurované pro použití seznamu názvů klastru nejsou monitorovány.
- Povolení přenosu zpráv z neaktivních front je volitelné. Vztahuje se na všechny fronty monitorované instancí ukázkového programu monitorování fronty klastru. Pokud je povolena pouze podmnožina monitorovaných front, je třeba přenos dvou instancí programu pro monitorování fronty klastru, který je povolen. Jeden ukázkový program má povolen přenos zpráv a druhý přenos zprávy je zakázán. Každá instance ukázkového programu potřebuje lokální frontu pro řídicí a odpovědní zprávy.
- IBM MQ vyrovnávání pracovní zátěže klastru bude standardně odesílat zprávy instancím klastrovaných front, které jsou umístěny ve stejném správci front, ke kterému je aplikace připojena. Tato volba musí být zakázána, pokud je lokální fronta neaktivní za následujících okolností:
  - Aplikace aplikací se připojí ke správcům front, kteří vlastní instance neaktivní fronty, která je monitorována.
  - Zprávy ve frontě jsou převáděny z neaktivních front do aktivních front.

  Lokální preference vyrovnávání pracovní zátěže ve frontě lze zakázat staticky, a to nastavením hodnoty CLWLUSEQ na hodnotu ANY. V těchto konfiguračních zprávách jsou lokální fronty distribuovány do lokálních a vzdálených instancí front pro vyvážení pracovní zátěže, a to i v případě, kdy existují lokální náročné aplikace. Ukázkový program monitorování fronty klastru může být také konfigurován tak, aby dočasně nastavil hodnotu **CLWLUSEQ** na ANY , zatímco fronta nemá žádné připojené spotřebitele, což vede k tomu, že lokální zprávy budou v době, kdy jsou tyto fronty aktivní, odesílány pouze lokální zprávy.
- Systém a aplikace IBM MQ nesmí používat produkt **CLWLPRTY** pro monitorované fronty nebo kanály, které mají být používány. Jinak by akce ukázkového programu monitorování fronty klastru na atributech fronty produktu **CLWLPRTY** mohly mít nepožadované účinky.
- Ukázkový program pro monitorování fronty klastru protokoluje informace o běhovém prostředí do sady souborů sestav. Požaduje se adresář pro uložení těchto sestav a ukázkový program pro monitorování fronty klastru musí mít oprávnění k zápisu do tohoto adresáře.

AMQSCLM: Příprava a spuštění ukázky
Ukázku monitorování fronty klastru lze lokálně spustit buď lokálně připojené ke správci front, nebo jako klient připojený prostřednictvím kanálu. Ukázka by měla být spuštěna vždy, když je správce front spuštěn, je-li spuštěn lokálně, lze jej nakonfigurovat jako službu správce front, aby automaticky spustil a zastavil ukázku s použitím správce front.

Než začnete

Před spuštěním ukázky monitorování fronty klastru je třeba provést následující kroky.

1. Vytvořte pracovní frontu v každém správci front za účelem interního použití ukázky.

   Každá instance ukázky potřebuje lokální neklastrovou frontu pro výlučné interní použití. Můžete zvolit název fronty. Příklad používá název `AMQSCLM.CONTROL.QUEUE`. V systému Windowsmůžete například vytvořit tuto frontu pomocí následujícího příkazu **MQSC** :

   ```
   DEFINE QLOCAL(AMQSCLM.CONTROL.QUEUE)
   ```

   Hodnoty **MAXDEPTH** a **MAXMSGL** můžete ponechat jako výchozí.

2. Vytvořte adresář pro protokoly chyb a informací o chybách.

   Ukázka vypíše diagnostické zprávy do souborů sestav. Je třeba vybrat adresář, do kterého chcete soubory uložit. V systému Windowsmůžete například vytvořit adresář pomocí následujícího příkazu:

   ```
   mkdir C:\AMQSCLM\rpts
   ```

   Soubory sestavy vytvořené v ukázce mají následující konvenci pojmenování:

   ```
   QmgrName.ClusterName.RPT0n.LOG
   ```

3. (Volitelné) Definujte ukázku monitorování fronty klastru jako službu IBM MQ .

   Chcete-li monitorovat fronty, musí být vzorek vždy spuštěn. Chcete-li se ujistit, že je ukázka monitorování fronty klastru vždy spuštěna, můžete definovat ukázku jako službu správce front. Definování ukázky jako služby znamená, že je spuštěn příkaz AMQSCLM při spuštění správce front. K definování ukázky monitorování fronty klastru jako služby IBM MQ můžete použít následující příklad.

   ```
   define service(AMQSCLM) +
         descr('Active Cluster Queue Message Distribution Monitor - AMQSCLM') +
         control(qmgr) +
         servtype(server) +
         startcmd('MQ_INSTALLATION_PATH\tools\c\samples\Bin\AMQSCLM.exe') +
         startarg('-m +QMNAME+ -c CLUSTER1 -q ABC* -r AMQSCLM.CONTROL.QUEUE -l
   c:\AMQSCLM\rpts') +
         stdout('C:\AMQSCLM\rpts\+QMNAME+.TSTCLUS.stdout.log') +
         stderr('C:\AMQSCLM\rpts\+QMNAME+.TSTCLUS.stderr.log')
   ```

Definice	Popis
service	Uvádí název služby. Můžete zvolit název služby.
descr	Uvádí textový popis služby.
control	Označuje, že služba se spustí a zastaví ve stejnou dobu jako správce front.
servtype	Označuje, že objekt služby serveru, což znamená pouze jednu instanci, může být proveden v daném okamžiku pro tohoto správce front.
startcmd	Uvádí umístění a jméno programu.
startarg	Určuje argumenty ukázky. Všimněte si použití volby *+ QMNAME +*. Název správce front se automaticky nahrazuje názvem správce front.
stdout	Plně kvalifikovaný název souboru, do kterého je přesměrován standardní výstup. Ukázka zapisuje do tohoto souboru pouze zprávy potvrzující, že vzorek byl ukončen. Ukázka to provede, protože soubor se standardním chybovým souborem již byl uzavřen v předchozí fázi ukázkového procesu ukončení.

Definice	Popis
stderr	Úplný název souboru, na který je přesměrován standardní výstup chybových hlášení. Ukázka zapisuje na standardní chybový soubor všechny chybové zprávy před ukončením ukázky.

Informace o této úloze

Tato úloha vám umožňuje spustit a zastavit ukázku monitorování fronty klastru různými způsoby. Umožňuje vám také spustit ukázku v režimu, který generuje soubory sestav obsahující statistické informace o monitorovaných frontách.

Ukázkový program lze spustit pomocí následujícího příkazu.

```
AMQSCLM -m QMgrName -c ClusterName (-q QNameMask| -f QListFile) -r MonitorQName
[-l ReportDir] [-t] [-u ActiveVal] [-i Interval] [-d] [-s] [-v]
```

V tabulce jsou uvedeny argumenty, které lze použít spolu s ukázkou monitorování fronty klastru, spolu s dalšími informacemi o každém z nich.

Argument	Proměnná	Další informace
-m	**QMgrName**	Správce front, který má být sledován.
-c	**ClusterName**	Klastr obsahující fronty určené k monitorování.
-q	**QNameMask**	Fronta, nebo fronty k monitorování. Koncový * monitoruje všechny fronty s názvy, které odpovídají nula nebo více koncových znacích.
-f	**QListFile**	Úplná cesta a název souboru obsahujícího seznam názvů front nebo masek názvů front, které mají být monitorovány. Soubor musí obsahovat jeden název fronty/masku na řádek. Můžete uvést **-q** nebo **-f**, ale ne obojí.
-r	**MonitorQName**	Lokální fronta je používána exkluzivně pro vzorek.
-l	**ReportDir**	Cesta k adresáři, do které mají být ukládány protokolované informační zprávy v sadě zalamování[10]soubory sestav.
-t		(Volitelné) Umožňuje přenos zpráv ve frontě z neaktivních lokálních front do aktivních front. Není-li tato možnost povolena, jsou do aktivních instancí fronty dynamicky směrovány pouze nové zprávy, které vstupují do klastru.
-u	**ActiveVal**	(Volitelné) Automaticky přepíná vlastnost **CLWLUSEQ** monitorované instance fronty na ANY , když je neaktivní, a na hodnotu **ActiveVal** , pokud je aktivní. **ActiveVal** může být LOCAL nebo QMGR. Není-li tento argument nastaven v systému, kde jsou aplikace připojeny ke stejnému správci front nebo je-li přenos zpráv povolen, pak monitorované fronty musí mít hodnotu **CLWLUSEQ** ANYnebo QMGR s hodnotou ANYsprávce front.
-i	**Interval**	(Volitelné) Časový interval v sekundách, ve kterém monitor kontroluje fronty. Výchozí hodnota je 300 sekund (5 minut).
-d		(Volitelné) Umožňuje další diagnostický výstup. Ladicí výstup může být užitečný při počáteční konfiguraci systému nebo při práci s ukázkovým kódem.
-s		(Volitelné) Povolí minimální statistický výstup na každý interval.
-v		(Volitelné) Chcete-li kromě souborů sestav také protokolovat informace o sestavě do produktu `standard out`.

Příklady seznamu argumentů:

```
-m QMGR1 -c CLUS1   -f c:\QList.txt -r CLMQ -l c:\amqsclm\rpts -s
-m QMGR2 -c CLUS1   -q ABC* -r CLMQ -l c:\amqsclm\rpts -i 600
-m QMGR1 -c CLUSDEV -q QUEUE.* -r CLMQ -l c:\amqsclm\rpts -t -u QMGR -d
```

Ukázkový soubor se seznamem front:

```
Q1
QUEUE.*
ABC
ABD
```

[10] Pro každou kombinaci správce front a fronty je vygenerován soubor protokolu pevné velikosti, který je při zaplnění přepsán. Modul protokolování vždy zapisuje do stejného souboru a uchovává také dvě předchozí verze souboru.

Postup

1. Spusťte ukázku monitorování fronty klastru. Ukázku můžete spustit jedním z následujících způsobů:
   - Použijte příkazový řádek s odpovídajícími oprávněními uživatele.
   - Pokud je ukázka konfigurována jako služba IBM MQ , použijte příkaz MQSC **START SERVICE** .

   Seznam argumentů je v obou případech stejný.

   Ukázka nespouští monitorování front po dobu 10 sekund od inicializace programu. Toto zpoždění umožňuje náročným aplikacím nejprve se připojit k monitorovaným frontám, čímž se zabrání zbytečným změnám aktivního stavu fronty.
2. Zastavte ukázku monitorování fronty klastru. Ukázka se automaticky zastaví při zastavení činnosti správce front, zastavení, uvádění do klidového stavu nebo v případě, že je připojení ke správci front přerušeno. Existují způsoby, jak zastavit ukázku bez ukončení správce front:
   - Nakonfigurujte lokální frontu použitou výhradně ukázkou k zakázání funkce Get.
   - Odešlete zprávu s **CorrelId** z `"STOP CLUSTER MONITOR\0\0\0\0"`do lokální fronty použité výlučně pro ukázku.
   - Ukončete ukázkový proces. To může vést ke ztrátě netrvalých zpráv přenášených do aktivních front. Může to také vést k tomu, že lokální fronta použitá vzorkovým vzorkem je otevřená po dobu několika sekund po ukončení. Tato situace zabrání okamžitému spuštění nové instance ukázky monitorování fronty klastru.

   Pokud byl vzorek spuštěn jako služba IBM MQ , **STOP SERVICE** nemá žádný efekt. Je možné použít jednu z metod ukončení popsaných jako konfigurovaný mechanizmus **STOP SERVICE** ve správci front.

Jak pokračovat dále

Zkontrolujte stav ukázky.

Je-li vykazování povoleno, můžete přezkoumat soubory sestavy pro stav. Použijte následující příkaz k přezkoumání nejnovějšího souboru sestavy:

```
QMgrName.ClusterName.RPT01.LOG
```

Chcete-li zkontrolovat starší soubory sestav, použijte následující příkazy:

```
QMgrName.ClusterName.RPT02.LOG
QMgrName.ClusterName.RPT03.LOG
```

Velikost souborů sestavy vzroste na maximální velikost přibližně 1 MB. Když se soubor RPT01 zaplní, vytvoří se nový soubor RPT01 . Starý soubor RPT01 je přejmenován na RPT02. RPT02 je přejmenován na RPT03. Původní RPT03 je vyřazeno.

Ukázka vytváří informační zprávy v následujících situacích:

- při spuštění
- při ukončení
- když je zaznačit frontu **ACTIVE** nebo **INACTIVE**
- když znovu požaduje zprávy z neaktivní fronty k aktivní instanci nebo instancím

Ukázka vytvoří chybovou zprávu *CLMnnnnE* k ohlášení problému, který vyžaduje pozornost.

Každých 30 minut vzorková zpráva uvádí průměrnou dobu zpracování na interval zjišťování a uplynulý čas zpracování. Tyto informace jsou uchovávány ve zprávě CLM0045I.

Jsou-li povoleny statistické zprávy **-s**, sestava ukázek vykazuje následující statistické informace o každé kontrole fronty:

- Čas potřebný ke zpracování front (v milisekundách)

- Počet zkontrolovaných front
- Počet aktivních/neaktivních změn provedených
- Počet přenesených zpráv

Tyto informace jsou uvedeny ve zprávě CLM0048I.

Soubory sestavy mohou rychle růst v režimu ladění a rychle se zalomit. V této situaci může být překročen limit velikosti 1 MB pro jednotlivé soubory.

AMQSCLM: Odstraňování problémů
Následující sekce obsahují informace o scénářích, které mohou být zjištěny při použití ukázky. Poskytují se informace o možných vysvětleních scénáře a o možnostech řešení tohoto scénáře.

Scénář: Produkt AMQSCLM se nespustí

Potenciální vysvětlení: Chybná syntaxe.

Akce: Zkontrolujte standardní chybový výstup pro správnou syntaxi

Potenciální vysvětlení: Správce front není k dispozici.

Akce: Zkontrolujte soubor sestavy pro ID zprávy CLM0010E.

Potenciální vysvětlení: Nelze otevřít nebo vytvořit soubor sestavy nebo soubory.

Akce: Během inicializace zkontrolujte chybové zprávy standardního výstupu chyb.

Scénář: AMQSCLM nemění frontu na AKTIVNÍ nebo NEAKTIVNÍ

Potenciální vysvětlení: Fronta se nenachází v seznamu front, které mají být monitorovány.

Akce: Zkontrolujte hodnoty parametrů **-q** a **-f** .

Potenciální vysvětlení: Fronta není lokální frontou ve správném klastru.

Akce: Zkontrolujte, zda je fronta lokální a ve správném klastru.

Potenciální vysvětlení: AMQSCLM není spuštěn pro tohoto správce front a klastru.

Akce: Spusťte AMQSCLM pro příslušného správce front a klastru.

Potenciální vysvětlení: Fronta je ponechána NEAKTIVNÍ, **CLWLPRTY** = 0, protože nemá žádné spotřebitele. Případně je ponechán AKTIVNÍ **CLWLPRTY** > =1, protože má alespoň 1 odběratele.

Akce: Zkontrolujte, zda jsou k frontě připojeny náročné aplikace.

Potenciální vysvětlení: Příkazový server správce front není spuštěn.

Akce: Zkontrolujte, zda soubory sestav nejsou chyby.

Scénář: Zprávy nejsou směrovány kolem NEAKTIVNÍ fronty.

Potenciální vysvětlení: Zprávy jsou vloženy přímo do správce front, který vlastní neaktivní frontu, a hodnota **CLWLUSEQ** fronty není ANYa argument **-u** se pro AMQSCLM nepoužívá.

Akce: Zkontrolujte hodnotu proměnné **CLWLUSEQ** příslušného správce front nebo zajistěte použití argumentu **-u** pro AMQSCLM.

Potenciální vysvětlení: U žádného správce front nejsou k dispozici žádné aktivní fronty. Zprávy jsou rovnoměrně rozloženy do všech neaktivních front, dokud se fronta nestane aktivní.

Akce: Zkontrolujte stav front ve všech správcích front.

Potenciální vysvětlení: Zprávy se umístí do jiného správce front v klastru do té, která vlastní neaktivní frontu, a aktualizovaná hodnota **CLWLPRTY** 0 není šířena do správce front pro uvedení aplikace.

Akce: Zkontrolujte, zda jsou spuštěny kanály klastru mezi monitorovaným správcem front a úplným správcem front úložiště. Zkontrolujte, zda jsou spuštěny kanály mezi umístěním správce front a správce

front úplného úložiště. Zkontrolujte protokoly chyb monitorovaných, umísťovačů a správců front úplného úložiště.

Potenciální vysvětlení: Instance vzdálených front jsou aktivní (CLWLPRTY=1), ale zprávy nelze směrovat na tyto instance fronty, protože odesílací kanál klastru z lokálního správce front není spuštěn.

Akce: Zkontrolujte stav odesílacích kanálů klastru z lokálního správce front do vzdáleného správce front nebo správců s aktivní instancí fronty.

Scénář: AMQSCLM nepřevádí zprávy z neaktivní fronty

Potenciální vysvětlení: Přenos zpráv není povolen (**-t**).

Akce: Ujistěte se, že přenos zpráv je povolen (**-t**).

Potenciální vysvětlení: Fronta se nenachází v seznamu front, které mají být monitorovány.

Akce: Zkontrolujte hodnoty parametrů **-q** a **-f** .

Potenciální vysvětlení: AMQSCLM není spuštěn pro tento správce front nebo pro jiné správce front v klastru, kteří vlastní instance stejné fronty.

Akce: Spusťte AMQSCLM.

Potenciální vysvětlení: Fronta má **CLWLUSEQ** = LOCAL nebo **CLWLUSEQ** = QMGR, a argument **-u** není nastaven.

Akce: Nastavte parametr **-u** , nebo změňte konfiguraci fronty nebo správce front na hodnotu ANY.

Potenciální vysvětlení: V klastru nejsou žádné aktivní instance fronty.

Akce: Zkontrolujte instance fronty s hodnotou **CLWLPRTY** 1, nebo vyšší.

Potenciální vysvětlení: Instance vzdálených front mají spotřebitele (**IPPROCS** > = 1), ale jsou neaktivní na těchto správcích front (**CLWLPRTY** = 0), protože AMQSCLM nemonitoruje tyto vzdálené instance.

Akce: Ujistěte se, že je v těchto správcích front spuštěn příkaz AMQSCLM a/nebo je fronta v seznamu front, které mají být monitorovány, kontrolou hodnot parametrů **-q** a **-f** .

Potenciální vysvětlení: Instance vzdálených front jsou aktivní (**CLWLPRTY** = 1), ale jsou považovány za neaktivní na lokálním správci front (**CLWLPRTY** = 0). Tato situace nastane, protože aktualizovaná hodnota produktu **CLWLPRTY** není šířena do tohoto správce front.

Akce: Ujistěte se, že jsou vzdálení správci front připojeni k alespoň jednomu správci front úložiště v klastru. Ujistěte se, že správci front úplného úložiště pracují správně. Zkontrolujte, zda jsou spuštěny kanály mezi správci front úplného úložiště a monitorovanými správci front.

Potenciální vysvětlení: Zprávy nejsou potvrzeny, proto nemohou být vyhledatelné.

Akce: Zkontrolujte, zda odesílající aplikace funguje správně.

Potenciální vysvětlení: AMQSCLM nemá přístup k lokální frontě, kde jsou zprávy zařazeny do fronty.

Akce: Zkontrolujte, zda je AMQSCLM spuštěn jako uživatel s dostatečnou autorizací pro přístup k frontě.

Potenciální vysvětlení: Příkazový server správce front není spuštěn.

Akce: Spusťte příkazový server správce front.

Možné vysvětlení: AMQSCLM rozpoznal chybu.

Akce: Zkontrolujte, zda soubory sestav nejsou chyby.

Potenciální vysvětlení: Instance vzdálených front jsou aktivní (CLWLPRTY=1), ale zprávy nelze přenést na tyto instance fronty, protože odesílací kanál klastru z lokálního správce front není spuštěn. K této akci je často připojeno varování CLM0030W v protokolu sestavy amqsclm.

Akce: Zkontrolujte stav odesílacích kanálů klastru z lokálního správce front do vzdáleného správce front nebo správců s aktivní instancí fronty.

ULW *Ukázkový program pro vyhledávání koncového bodu připojení (CEPL)*

Ukázka Vyhledávání koncového bodu připojení produktu IBM MQ poskytuje jednoduchý, ale účinný modul ukončení, který nabízí uživatelům produktu IBM MQ způsob, jak načíst definice připojení z úložiště LDAP, jako je například produkt Tivoli Directory Server.

Tivoli Directory Server 6.3 Klient musí být instalován, aby bylo možné používat CEPL.

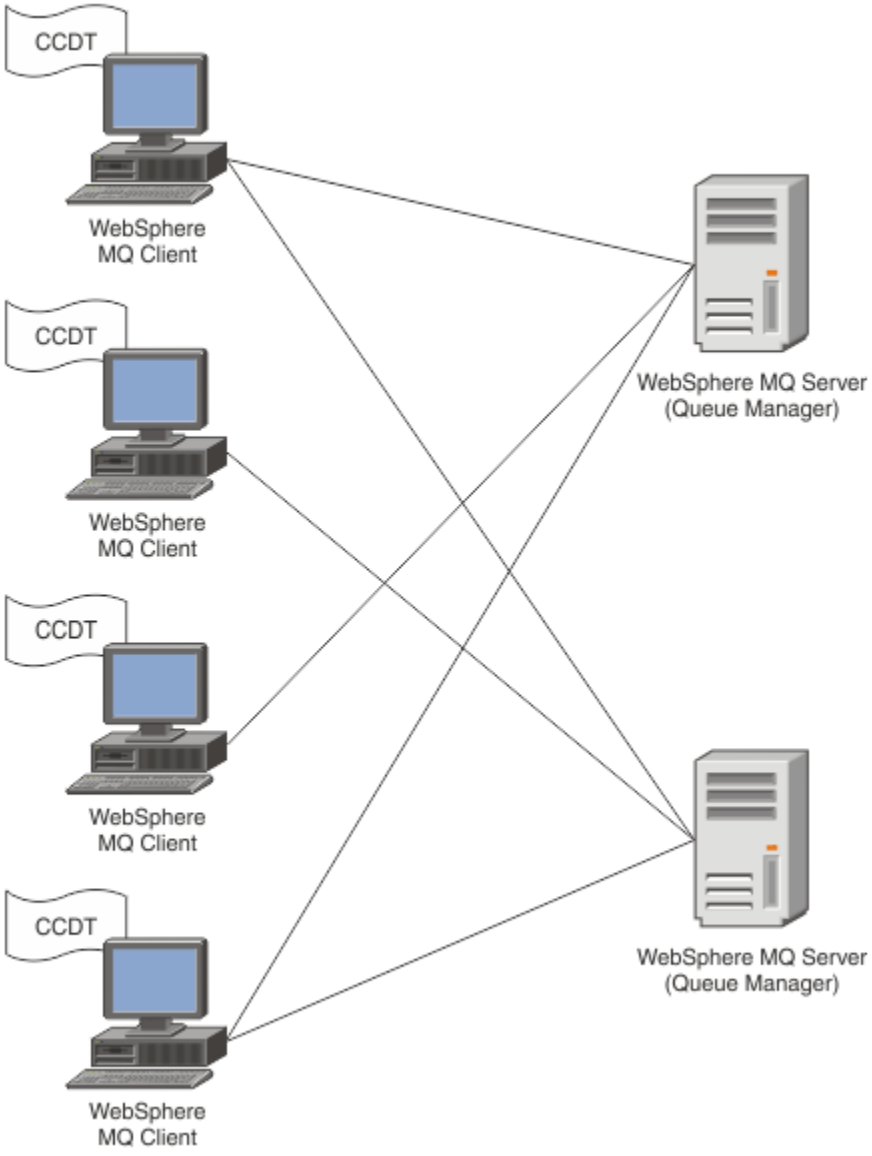
Pro použití této ukázky je vyžadováno pracovní znalosti administrace produktu IBM MQ na podporovaných platformách.

Windows Solaris Linux AIX *Úvod*

Nakonfigurujte globální úložiště, například adresář LDAP (Lightweight Directory Access Protocol), chcete-li uložit definice připojení klienta pro podporu údržby a administrace.

Pomocí aplikace klienta IBM MQ lze navázat spojení se správcem front prostřednictvím tabulky CCDT (Client Connection Definition Table).

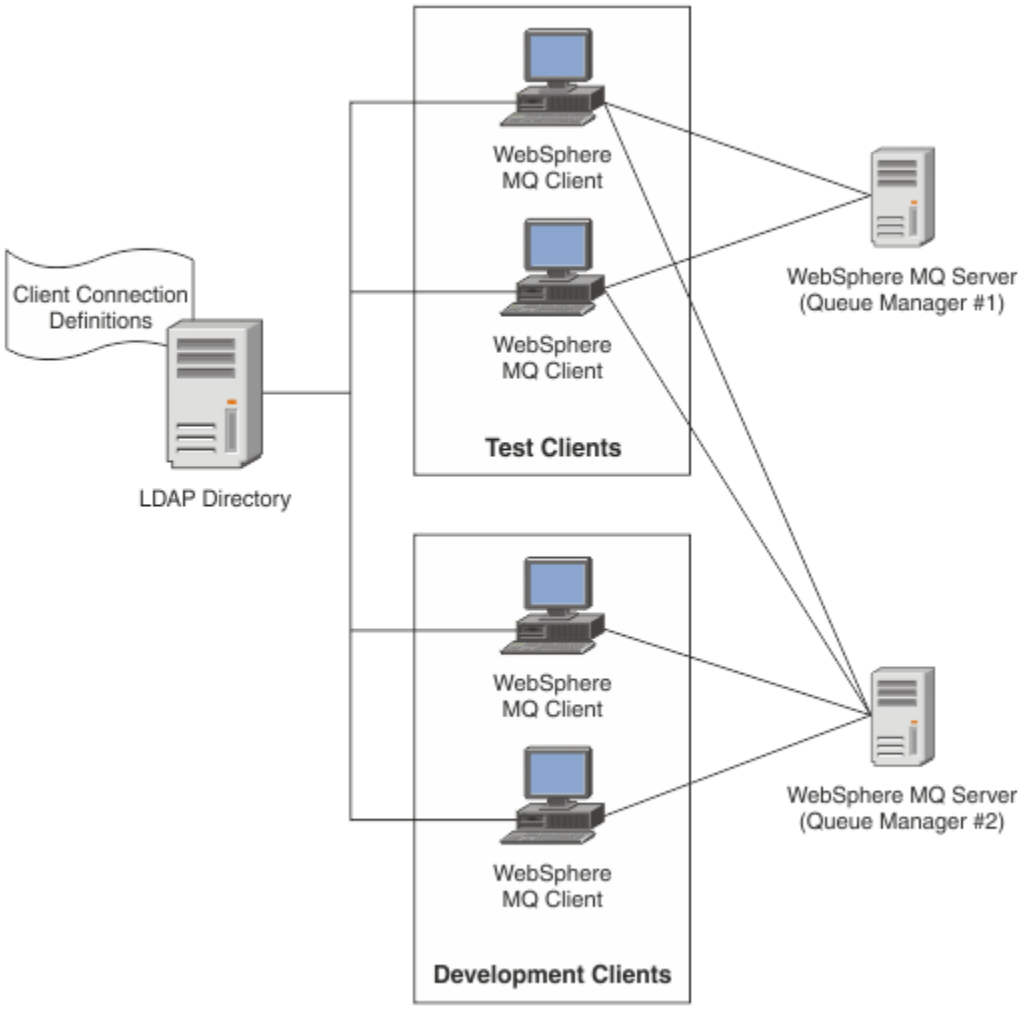
Nástroje CCDT se vytvářejí prostřednictvím standardního rozhraní pro administraci produktu IBM MQ MQSC. Uživatel musí být připojen ke správci front, aby bylo možné vytvořit definice připojení klienta, i když data obsažená v definici nejsou omezena na správce front. Generovaný soubor CCDT musí být ručně distribuován mezi klientskými počítači a aplikacemi.



Soubor CCDT musí být distribuován na každého klienta IBM MQ . V případě, že tisíce klientů mohou existovat buď lokálně, nebo globálně, brzy by bylo obtížné udržovat a spravovat. Je zapotřebí flexibilnějšího přístupu, který pomáhá zajistit, aby každý klient měl k dispozici správné definice klienta.

Jednou z takových přístupů je uložit definice připojení klienta do globálního úložiště, jako je například adresář LDAP (Lightweight Directory Access Protocol). Adresář LDAP může také poskytovat další funkce zabezpečení, indexace a vyhledávání, čímž umožňuje každému klientovi přístup pouze k těm definicím připojení, které se jich týkají.

Adresář LDAP lze nakonfigurovat tak, aby byly k dispozici pouze určité definice pro určité skupiny uživatelů. Testovací klienti mohou například přistupovat k oběma správci front #1 i k produktu #2, zatímco klienti Development Clients mohou přistupovat pouze k produktu Queue Manager #2 .



Výstupní modul může vyhledat úložiště LDAP, například IBM Tivoli Directory Server, chcete-li načíst definice kanálu. Pomocí těchto definic připojení může klientská aplikace produktu IBM MQ navázat připojení ke správci front.

Výstupní modul je předpřipojovacím uživatelským modulem, který umožňuje získat definici kanálu během volání MQCONN/MQCONNX z úložiště LDAP.

Výstupní modul a schéma mohou být implementovány pomocí:

- Zákazníci, kteří již vybudovali základ dovednosti s využitím existující technologie založené na souborech CCDT a chtějí snížit náklady na administraci a distribuci.
- Existující zákazníci, kteří již používají svou vlastní proslušnou technologii pro distribuci definic připojení klientů.

- Noví nebo stávající zákazníci, kteří momentálně nepoužívají žádný typ řešení připojení klienta a chtějí využívat funkce nabízené produktem IBM MQ.
- Noví nebo stávající zákazníci, kteří chtějí přímo používat nebo ladit svůj model systému zpráv vložený s libovolnou aktuální obchodní architekturou LDAP.

ULW *Podporovaná prostředí*

Před spuštěním ukázky vyhledání koncového bodu připojení ověřte, že máte podporovaný operační systém a odpovídající software.

Ukázkový program pro IBM MQ Vyhledávání koncového bodu připojení vyžaduje následující software:

- IBM WebSphere MQ 7.0nebo novější
- Tivoli Directory Server 6.3 Client nebo novější

Podporované operační systémy:

1. Windows Windows (7/8/2008/2012)
2. Solaris Solaris (SPARC a x86-64)
3. AIX AIX
4. Linux Linux
   - RHEL v4 a v5 v systému System p
   - SUSE v9 a v10 na System p
   - RHEL v4 a v5 System x 32 bitů a 64 bitů
   - SUSE v9 a v10 System x 32 bit a 64 bit
5. HP-UX HP IA64.

Poznámka: Ukázka není k dispozici pro následující platformy:

- z/OS z/OS
- IBM i IBM i
- HP-UX HP PA-RISHO

ULW *Instalace a konfigurace*

Instalace a konfigurace výstupního modulu a schématu koncového bodu připojení.

Instalace výstupního modulu

Během instalace produktu IBM MQje modul uživatelské procedury instalován pod `tools/samples/c/preconnexit/bin`. Pro 32bitové platformy musí být výstupní modul zkopírován do produktu `exit/`*`installation_name`*`/` dříve, než jej lze použít. Pro 64bitové platformy musí být výstupní modul zkopírován do exit64/*název_instalace*/před tím, než bude možné jej použít.

Instalace schématu koncového bodu připojení

Uživatelská procedura používá schéma koncového bodu připojení, `ibm-amq.schema`. Před použitím uživatelské procedury musí být soubor schématu importován do libovolného serveru LDAP. Po importu schématu musí být přidány hodnoty pro atributy.

Zde je příklad pro import schématu koncového bodu připojení. Příklad předpokládá, že se používá IBM Tivoli Directory Server (ITDS).

- Ujistěte se, že IBM Tivoli Directory Server běží, pak zkopírujte nebo FTP soubor `ibm-amq.schema` na server ITDS.

- Na serveru ITDS zadejte následující příkaz pro instalaci schématu do úložiště ITDS, kde *LDAP ID* a *heslo LDAP* jsou kořenové DN a heslo pro server LDAP:

  ```
  ldapadd -D "LDAP ID" -w "LDAP password" -f ibm-amq.schema
  ```

- V příkazovém okně zadejte následující příkaz nebo použijte nástroj třetí strany k procházení schématu pro ověření:

  ```
  ldapsearch objectclass=ibm-amqClientConnection
  ```

Další podrobnosti o importu souboru schématu najdete v dokumentaci k serveru LDAP.

Konfigurace

Do konfiguračního souboru klienta musí být přidána nová sekce s názvem PreConnect , například `mqclient.ini`. Sekce PreConnect obsahuje následující klíčová slova:

Modul: Název modulu, který obsahuje kód ukončení rozhraní API. Pokud toto pole obsahuje úplnou cestu k modulu, použije se tak, jak je. Jinak se prohledá složka `exit` nebo `exit64` v instalaci produktu IBM MQ .

Funkce: Název funkčního vstupního bodu do knihovny, která obsahuje výstupní kód PreConnect . Definice funkce dodržuje prototyp MQ_PRECONNECT_EXIT.

Data: URI úložiště LDAP obsahujícího definice kanálů.

Následující úsek kódu je příkladem změn požadovaných v souboru `mqclient.ini` .

```
PreConnect:
Module=amqlcelp
Function=PreConnectExit
Data=ldap://myLDAPServer.com:389/cn=wmq,ou=ibm,ou=com
Sequence=1
```

ULW *Přehled o ukončení a schématu*

Syntaxe a parametry použité k vytvoření připojení ke správci front.

IBM MQ 9.0 definuje následující syntaxi pro vstupní bod v modulu uživatelské procedury.

```
void MQENTRY MQ_PRECONNECT_EXIT ( PMQNXP pExitParms
                                , PMQCHAR pQMgrName
                                , PPMQCNO ppConnectOpts
                                , PMQLONG pCompCode
                                , PMQLONG pReason)
```

Během provádění volání MQCONN/X klient IBM MQ C načte výstupní modul obsahující implementaci syntaxe funkce. Pak vyvolá funkci ukončení, aby načetla definice kanálu. Načtené definice kanálu se poté používají k navázání spojení se správcem front.

Parametry

Parametry příkazu pExit

Typ: PMQNXP I/O

Struktura konfiguračního parametru PreConnection . Tato struktura je přidělována a udržována volajícím pro ukončení.

```
struct tagMQNXP
{
  MQCHAR4    StrucId;             /* Structure identifier */
  MQLONG     Version;             /* Structure version number */
  MQLONG     ExitId;              /* Type of exit */
  MQLONG     ExitReason;          /* Reason for invoking exit */
  MQLONG     ExitResponse;        /* Response from exit */
  MQLONG     ExitResponse2;       /* Secondary response from exit */
```

```
  MQLONG     Feedback;            /* Feedback code (reserved) */
  MQLONG     ExitDataLength;      /* Exit data length */
  PMQCHAR    pExitDataPtr;        /* Exit data */
  MQPTR      pExitUserAreaPtr;    /* Exit user area */
  PMQCD *    ppMQCDArrayPtr;      /* Array of pointers to MQCDs */
  MQLONG     MQCDArrayCount;      /* Number of entries found */
  MQLONG     MaxMQCDVersion;      /* Maximum MQCD version */
};
```

Název pQMgr

Typ: PMQCHAR vstupní/výstupní

Název správce front. Na vstupu je tento parametr řetězec filtru dodávaný do volání rozhraní API MQCONN prostřednictvím parametru **QMgrName** . Toto pole může být prázdné, explicitní nebo obsahovat určité zástupné znaky. Pole je změněno uživatelskou procedurou. Tento parametr má hodnotu NULL , když je uživatelská procedura volána s MQXR_TERM.

ppConnectOpts

Typ: ppConnectOpts vstup/výstup

Volby, které řídí akci MQCONNX. Jedná se o ukazatel na strukturu voleb připojení MQCNO, která řídí akci volání rozhraní API MQCONN. Tento parametr má hodnotu NULL , když je uživatelská procedura volána s MQXR_TERM. Klient MQI vždy poskytuje strukturu MQCNO pro ukončení i v případě, že ji aplikace původně neposkytovala. Pokud aplikace poskytuje strukturu MQCNO, klient vytvoří duplikát a předá jej do uživatelské procedury, kde je upravena. Klient si zachová vlastnictví objektu MQCNO. MQCD, na které odkazuje objekt MQCNO, má přednost před jakoukoli definicí připojení poskytnutou prostřednictvím pole. Klient používá strukturu MQCNO k připojení ke správci front a ostatní jsou ignorovány.

Kód pComp

Typ: PMQLONG vstupní/výstupní

Kód dokončení. Ukazatel na MQLONG, který přijímá kód dokončení ukončení. Musí se jednat o jednu z následujících hodnot:

- MQCC_OK -Úspěšné dokončení.
- MQCC_WARNING -Varování (částečné dokončení)
- MQCC_FAILED -Volání se nezdařilo.

pReason

Typ: PMQLONG vstupní/výstupní

Kód určující kvalifikaci pCompCode. Ukazatel na hodnotu MQLONG, která přijímá kód příčiny ukončení. Je-li kód dokončení MQCC_OK, jediná platná hodnota je: MQRC_NONE -(0, x '000') Ne důvod k vytvoření sestavy.

Je-li kód dokončení MQCC_FAILED nebo MQCC_WARNING, může funkce uživatelské procedury nastavit pole s kódem příčiny na jakoukoli platnou hodnotu MQRC_ *.

ULW *Informace o kontextu služby LDAP produktu MQ*

Uživatelská procedura používá následující strukturu dat pro kontextové informace.

MQNLDAPCTX

Struktura MQNLDAPCTX má následující prototyp C.

```
typedef struct tagMQNLDAPCTX MQNLDAPCTX;
typedef MQNLDAPCTX MQPOINTER PMQNLDAPCTX;

struct tagMQNLDAPCTX
{
  MQCHAR4      StrucId;          /* Structure identifier */
  MQLONG       Version;          /* Structure version number */
  LDAP *       objectDirectory   /* LDAP Instance */
  MQLONG       ldapVersion;      /* Which LDAP version to use? */
  MQLONG       port;             /* Port number for LDAP server*/
  MQLONG       sizeLimit;        /* Size limit */
  MQBOOL       ssl;              /* SSL enabled? */
  MQCHAR *     host;             /* Hostname of LDAP server */
  MQCHAR *     password;         /* Password of LDAP server */
```

```
  MQCHAR *     searchFilter;     /* LDAP search filter */
  MQCHAR *     baseDN;           /* Base Distinguished Name */
  MQCHAR *     charSet;          /* Character set */
};
```

Windows Solaris Linux AIX *Ukázkový kód pro sestavení uživatelské procedury pro vyhledání koncového bodu připojení*
Můžete použít úseky kódu ukázky pro kompilaci zdroje na systémech AIX, Linux, Solarisa Windows.

Kompilace zdroje

Můžete kompilovat zdroj s libovolní knihovnami klienta LDAP, například IBM Tivoli Directory Server 6.3 Knihovny klienta. Tato dokumentace předpokládá, že používáte knihovny klienta produktu Tivoli Directory Server 6.3 .

Poznámka: Knihovna uživatelské procedury před připojením je podporována u následujících serverů LDAP:

- IBM Tivoli Directory Server 6.3
- Novell eDirectory 8.2

Následující úseky kódu popisují, jak zkompilovat uživatelské procedury:

Windows **Kompilování uživatelské procedury na platformě Windows**
Při kompilaci zdroje ukončení můžete použít následující úsek kódu:

```
CC=cl.exe
LL=link.exe
CCARGS=/c /I. /DWIN32 /W3 /DNDEBUG /EHsc /D_CRT_SECURE_NO_DEPRECATE /Zl

# The libraries to include
LDLIBS=Ws2_32.lib Advapi32.lib libibmldapstatic.lib libibmldapdbgstatic.lib \
kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.lib \
shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib msvcrt.lib

OBJS=amqlcel0.obj

all: amqlcelp.dll

amqlcelp.dll: $(OBJS)
        $(LL) /OUT:amqlcelp.dll /INCREMENTAL /NOLOGO /DLL /SUBSYSTEM:WINDOWS /MACHINE: X86 \
              /DEF:amqlcelp.def $(OBJS) $(LDLIBS) /NODEFAULTLIB:msvcrt.lib

# The exit source
amqlcel0.obj: amqlcel0.c
$(CC) $(CCARGS) $*.c
```

Poznámka: Pokud používáte knihovny klienta produktu IBM Tivoli Directory Server 6.3 , které jsou kompilovány s kompilátorem produktu Microsoft Visual Studio 2003 , může dojít k varováním při kompilaci knihoven klienta produktu IBM Tivoli Directory Server 6.3 s kompilátorem Microsoft Visual Studio 2012nebo novější.

Solaris Linux AIX **Kompilace uživatelské procedury na AIX, Linuxnebo Solaris**
Následující úsek kódu je pro kompilaci ukončovacího zdroje na serveru Linux. Některé volby kompilátoru se mohou lišit od AIX nebo Solaris.

```
#Make file to build exit
CC=gcc

MQML=/opt/mqm/lib
MQMI=/opt/mqm/inc
TDSI=/opt/ibm/ldap/V6.3/include
XFLAG=-m32

TDSL=/opt/ibm/ldap/V6.3/lib
```

Produkt IBM Tivoli Directory Server je dodáván jak statických, tak i dynamicky propojovacím knihovnami, ale můžete použít pouze jeden typ knihovny. Tento skript předpokládá, že používáte statické knihovny.

```
#Use static libraries.
LDLIBS=-L$(TDSL) -libibmldapstatic

CFLAGS=-I. -I$(MQMI) -I$(TDSI)

all:amqlcepl

amqlcepl: amqlcel0.c
$(CC) -o cepl amqlcel0.c -shared -fPIC $(XFLAG) $(CFLAGS) $(LDLIBS)
```

ULW *Vyvolání výstupního modulu PreConnect*

Modul uživatelské procedury PreConnect lze vyvolat se třemi různými kódy příčiny: kódem příčiny MQXR_INIT pro inicializaci a zavedením připojení k serveru LDAP, kódem příčiny MQXR_PRECONNECT pro načítání definic kanálů ze serveru LDAP nebo kódem příčiny MQXR_TERM při čištění uživatelské procedury.

FUNKCE MQXR_INIT

Ukončení je vyvoláno s kódem příčiny MQXR_INIT pro inicializaci a navázání spojení se serverem LDAP.

Před voláním funkce MQXR_INIT je pole pExitDataPtr struktury MQNXP naplněno atributem Data ze stanzy PreConnect v rámci souboru `mqclient.ini` (tj. LDAP).

Adresa URL protokolu LDAP se skládá alespoň z protokolu, názvu hostitele, čísla portu a základního rozlišujícího názvu pro hledání. Uživatelská procedura analyzuje adresu URL protokolu LDAP obsaženou v poli pExitDataPtr , přidělí strukturu kontextu vyhledávání LDAP MQNLDAPCTX a naplní ji odpovídajícím způsobem. Adresa této struktury je uložena v poli pExitUserAreaPtr. Selhání při správné analýze výsledků adresy URL protokolu LDAP v chybě MQCC_FAILED.

V tomto bodě se uživatelská procedura připojí k serveru LDAP pomocí parametrů **MQNLDAPCTX** a připojí se k ní. Výsledné popisovače rozhraní LDAP API jsou také uloženy v této struktuře.

PŘIPOJENÍ MQXR_PRECONNECT

Výstupní modul je vyvolán s kódem příčiny MQXR_PRECONNECT pro načítání definic kanálů ze serveru LDAP.

Ukončení prohledává server LDAP pro definice kanálu odpovídající danému filtru. Pokud **QMgrNameparameter** obsahuje určité jméno správce front, vyhledávání vrátí všechny definice kanálu, pro které hodnota atributu LDAP **ibm-amqQueueManagerName** odpovídá danému názvu správce front.

Pokud je argument **QMgrName** '*' nebo ' ' (mezera), potom vyhledávání vrátí všechny definice kanálu, pro které je atribut koncového bodu **ibm-amqIsClientDefault Connection** nastaven na hodnotu TRUE.

Po úspěšném hledání procedura připraví jeden nebo pole definic MQCD a vrátí se zpět volajícímu.

VÝRAZ MQXR_

Ukončení je vyvoláno s tímto kódem příčiny, když se má uživatelská procedura vyčistit. Při tomto čištění se ukončení odpojí od serveru LDAP a uvolní veškerou přidělenou a udržovanou paměť, včetně struktury MQNLDAPCTX, pole ukazatele a každého odkazu MQCD, na které se odkazuje. Všechna ostatní pole jsou nastavena na výchozí hodnoty. Parametry ukončení **pQMgrName** a **ppConnectOpts** nejsou během ukončení s kódem příčiny MQXR_TERM nepoužívané a mohou mít hodnotu NULL.

Související informace

stanza PreConnect konfiguračního souboru klienta

ULW *Schémata LDAP*

Data připojení klienta jsou uložena v globálním úložišti, označeném jako adresář LDAP (Lightweight Directory Access Protocol). Klient IBM MQ používá adresář LDAP k získání definic připojení. Struktura

definic připojení klienta IBM MQ v rámci adresáře LDAP je známá jako schéma LDAP. Schéma LDAP je kolekce definic typů atributů, definic tříd objektů a dalších informací, které server používá k určení, zda se shoda filtru nebo atributu hodnoty shoduje s atributy záznamu, a zda se má povolit, přidat a upravit operace.

Ukládání dat do adresáře LDAP

Definice připojení klienta jsou umístěny pod specifickou větví v adresářovém stromu známém jako přípojný bod. Podobně jako všechny ostatní uzly v rámci adresáře LDAP má připojovací bod k sobě přidružený rozlišující název (DN). Tento uzel můžete použít jako výchozí bod pro všechny dotazy, které vytvoříte v adresáři. Použijte filtrování při dotazu na adresář LDAP k vrácení podmnožiny definic připojení klienta. Přístup k podstromům můžete omezit na základě oprávnění udělených v jiných částech adresářového stromu-například pro uživatele, oddělení nebo skupiny.

Definování vlastních atributů a tříd

Uložte definici kanálu klienta úpravou schématu LDAP. Všechny definice dat LDAP vyžadují objekty a atributy. Objekty a atributy jsou identifikovány identifikátorem OID (Object Identifier), který jednoznačně identifikuje objekt nebo atribut. Všechny třídy v rámci schématu LDAP dědí buď přímo, nebo nepřímo od horního objektu. Objekt definice kanálu klienta obsahuje atributy horního objektu. Všechny definice dat LDAP vyžadují objekty a atributy:

- Definice objektů jsou kolekce atributů LDAP.
- Atributy jsou datové typy LDAP.

Popis jednotlivých atributů a jejich mapování na běžné vlastnosti produktu IBM MQ je popsán v tématu Atributy LDAP.

ULW *Atributy LDAP*

Definované atributy LDAP jsou specifické pro IBM MQ a mapují se přímo na vlastnosti připojení klienta.

Atributy řetězce adresáře klienta klienta IBM MQ

Atributy znakového řetězce s jejich mapováním na vlastnosti produktu IBM MQ jsou vypsány v následující tabulce. Atributy mohou obsahovat hodnoty directoryString (UTF-8 encoded Unicode, to je proměnná byte encoding systému, která obsahuje syntaxi IA5/ASCII jako dílčí sadu). Syntaxe je uvedena jeho identifikačním číslem objektu (OID).

Tabulka 155. Atributy řetězce adresáře kanálu klienta produktu IBM MQ

atribut LDAP	Popis	IBM MQ Vlastnost
CN	Obecný název sestávající z názvu kanálu a názvu definujícího správce front.	
ibm-amqChannelNázev	Název definice kanálu.	CHANNEL
ibm-amqConnectionNázev	Identifikátor komunikačního připojení.	CONNAME
ibm-amqDescription	Popis kanálu.	DESCR
ibm-amqLocal	Lokální komunikační adresa kanálu.	LOCLADDR
ibm-amqModeNázev	Název režimu LU 6.2	MODENAME
ibm-amqPassword	Heslo, které lze použít.	PASSWORD
ibm-amqQueueManagerName	Název správce front nebo skupiny správců front, ke které může klientská aplikace klienta IBM MQ požadovat připojení.	QMNAME
ibm-amqSecurityExitUserData	Uživatelská data, která jsou předána uživatelské proceduře pro zabezpečení zprávy.	SCYDATA

Tabulka 155. Atributy řetězce adresáře kanálu klienta produktu IBM MQ (pokračování)		
atribut LDAP	**Popis**	**IBM MQ Vlastnost**
ibm-amqSecurityExitName	Název ukončovacího programu, který má být spuštěn uživatelskou procedurou zabezpečení kanálu.	SCYEXIT
ibm-amqSslCipherSpec	Jedna CipherSpec pro připojení TLS.	SSLCIPH
ibm-amqSslPeerName	Zkontroluje rozlišující název (DN) certifikátu od partnerského správce front nebo klienta na druhém konci kanálu produktu IBM MQ .	SSLPEER
ibm-amqTransactionProgramName	Název transakčního programu.	TPNAME
ibm-amqUserID	Jméno uživatele, které má být použito agentem MCA při pokusu o zahájení zabezpečené relace SNA se vzdáleným agentem MCA.	USERID

Atributy celého čísla připojení klienta IBM MQ

Atributy s předdefinovanými hodnotami (například výčtový typ) jsou uloženy jako standardní celá čísla. Tyto hodnoty jsou uloženy v adresáři LDAP jako celočíselné hodnoty, a ne pomocí přidruženého názvu konstanty.

Tabulka 156. Celočíselné atributy adresáře kanálu klienta IBM MQ		
atribut LDAP	**Popis**	**IBM MQ Vlastnost**
Afinita ibm-amqConnection	Určuje, zda klientské aplikace, které se připojují vícekrát přes stejné jméno správce front, používají stejný kanál klienta.	AFFINITY
ibm-amqClientChannelWeight	Váhový faktor, který ovlivňuje použitou definici kanálu připojení klienta.	CLNTWGHT
ibm-amqHeartBeatInterval	Přibližný čas mezi toky synchronizačních signálů předávanými z agenta kanálu zpráv v případě, kdy se v přenosové frontě nenacházejí žádné zprávy.	HBINT
ibm-amqKeepAliveInterval	Hodnota časového limitu pro kanál.	KAINT
ibm-amqMaximumMessageLength	Maximální délka zprávy, kterou lze přenést v kanálu.	MAXMSGL
ibm-amqSharingKonverzace	Maximální počet konverzací, které sdílejí každou instanci kanálu TCP/IP.	SHARECNV
ibm-amqTransportTyp	Typ transportu, který má být použit.	TRPTYPE

Atribut logického atributu kanálu klienta IBM MQ

Tento logický atribut není mapován na žádnou vlastnost IBM MQ . Syntaxe tohoto atributu označuje logickou hodnotu.

Tabulka 157. Atribut logického atributu kanálu klienta IBM MQ	
atribut LDAP	**Popis**
ibm-amqIsClientDefault	Tento logický atribut je definován pro vyřešení problému při hledání položek, jejichž atribut ibm-amqQueueManagerName nebyl definován.

Atributy seznamu kanálů klienta IBM MQ

Vlastnosti produktu IBM MQ se ukládají v rámci adresáře LDAP jako atribut seznamu s jedinou hodnotou a čárkami jako oddělovači. Atributy se definují stejným způsobem jako ostatní atributy řetězce adresáře. Atributy seznamu spolu s jejich mapováním na vlastnosti produktu IBM MQ jsou popsány v následující tabulce.

Tabulka 158. Atributy seznamu kanálů klienta IBM MQ

atribut LDAP	Popis	IBM MQ Vlastnost
ibm-amqHeaderKomprese	Seznam metod komprese dat záhlaví podporovaných kanálem.	COMPHDR
ibm-amqMessageCompression	Seznam technik komprese dat zpráv podporovaných kanálem.	COMPMSG
ibm-amqSendExitUserData	Uživatelská data, která jsou předána uživatelské proceduře pro odeslání zprávy.	SENDDATA
ibm-amqSendExitUserNázev	Název ukončovacího programu, který má být spuštěn uživatelskou procedurou odeslání kanálu.	SENDEXIT
ibm-amqReceiveExitUserData	Uživatelská data, která jsou předána uživatelské proceduře pro přijetí zprávy.	RCVDATA
ibm-amqReceiveExitName	Název uživatelského ukončovacího programu, který má být spuštěn uživatelskou procedurou kanálu pro přijetí zprávy.	RCVEXIT

ULW *Obecný název*

Obecný název (CN) se skládá z názvu kanálu a definování názvu správce front.

Jedná se o předexistující atribut.

Formát KN je:

```
CN=CHANNEL_NAME(DEFINING_Q_MGR_NAME)
```

Příklad:

```
CN=TC1(QM_T1)
```

Pro tento atribut můžete zadat pouze jednu hodnotu.

Tento atribut je řetězcový atribut a hodnoty nerozlišují velikost písmen. Shoda podřetězce je ignorována. Porovnávání podřetězce je porovnávací pravidlo použité v podschématu, které určuje chování atributu ve filtru vyhledávání, s použitím podřetězce (například CN=jim *, kde CN je atribut) a obsahuje jeden nebo více zástupných znaků.

ULW *Název ibm-amqChannel*

Tento atribut určuje název definice kanálu.

Tento atribut má jedinou řetězcovou hodnotu s maximálně 20 znaky, které nejsou citlivé na velikost písmen. Není to již existující atribut.

Shoda podřetězce je ignorována. Porovnávání podřetězce je porovnávací pravidlo použité v dílčím schématu, které určuje chování atributu ve filtru vyhledávání, pomocí podřetězce a obsahuje jeden nebo více zástupných znaků.

ULW *ibm-amqDescription*
Tento atribut LDAP poskytuje popis kanálu.

Tento atribut má jedinou řetězcovou hodnotu s maximálně 64 bajty, které nejsou citlivé na velikost písmen. Není to již existující atribut.

Shoda podřetězce je ignorována. Porovnávání podřetězců je odpovídající pravidlo použité v dílčím schématu, které určuje chování atributu ve filtru vyhledávání.

ULW *Název ibm-amqConnection*
Tento atribut LDAP je identifikátor komunikačního připojení. Určuje konkrétní komunikační spojení, které má tento kanál používat.

Tento atribut má jedinou řetězcovou hodnotu s maximálně 264 znaky, které nejsou citlivé na velikost písmen. Není to již existující atribut.

Shoda podřetězce je ignorována. Porovnávání podřetězců je odpovídající pravidlo použité v dílčím schématu, které určuje chování atributu ve filtru vyhledávání.

ULW *Adresa ibm-amqLocal*
Tento atribut určuje adresu lokální komunikace pro kanál.

Tento atribut má jedinou řetězcovou hodnotu s maximálně 48 znaky, které nejsou citlivé na velikost písmen. Není to již existující atribut.

Shoda podřetězce je ignorována. Porovnávání podřetězců je odpovídající pravidlo použité v dílčím schématu, které určuje chování atributu ve filtru vyhledávání.

ULW *Název ibm-amqMode*
Tento atribut je určen pro použití s připojeními LU 6.2. Poskytuje další definici charakteristik relace připojení, když se provádí alokace komunikační relace.

Tento atribut má jedinou řetězcovou hodnotu přesně 8 znaků, které nejsou citlivé na velikost písmen. Není to již existující atribut.

Shoda podřetězce je ignorována. Porovnávání podřetězců je odpovídající pravidlo použité v dílčím schématu, které určuje chování atributu ve filtru vyhledávání.

ULW *ibm-amqPassword*
Tento atribut LDAP určuje heslo, které může být použito agentem MCA při pokusu o zahájení zabezpečené relace LU 6.2 se vzdáleným agentem MCA.

Tento atribut má jedinou celočíselnou hodnotu o maximální délce 12 číslic. Není to již existující atribut.

ULW *ibm-amqQueueManagerName*
Tento atribut určuje název správce front nebo skupiny správců front, ke které může klientská aplikace klienta IBM MQ požadovat připojení.

Tento atribut má jedinou řetězcovou hodnotu s maximálně 48 znaky, které nejsou citlivé na velikost písmen. Není to již existující atribut.

Shoda podřetězce je ignorována. Porovnávání podřetězců je odpovídající pravidlo použité v dílčím schématu, které určuje chování atributu ve filtru vyhledávání.

Související odkazy
“ibm-amqIsClientDefault” na stránce 1135
Tento logický atribut řeší problém vyhledávání položek, kde atribut ibm-amqQueueManagerName nebyl definován.

ULW *Data ibm-amqSecurityExitUser*
Tento atribut LDAP určuje uživatelská data, která jsou předána uživatelské proceduře pro zabezpečení zprávy.

Tento atribut má jedinou řetězcovou hodnotu s maximálně 999 znaky, které nejsou citlivé na velikost písmen. Není to již existující atribut.

Shoda podřetězce je ignorována. Porovnávání podřetězců je odpovídající pravidlo použité v dílčím schématu, které určuje chování atributu ve filtru vyhledávání.

ULW *ibm-amqSecurityExitName*
Tento atribut LDAP uvádí název ukončovacího programu, který má být spuštěn uživatelskou procedurou zabezpečení kanálu.

Ponechte prázdné, není-li v platnosti žádná uživatelská procedura zabezpečení kanálu.

Tento atribut má jedinou řetězcovou hodnotu s maximálně 999 znaky, které nejsou citlivé na velikost písmen. Tento atribut není předukončující.

Shoda podřetězce je ignorována. Porovnávání podřetězců je odpovídající pravidlo použité v dílčím schématu, které určuje chování atributu ve filtru vyhledávání.

ULW *ibm-amqSslCipherSpec*
Tento atribut LDAP určuje jednu CipherSpec pro připojení TLS.

Tento atribut má jedinou řetězcovou hodnotu o maximální délce 32 znaků, které nejsou citlivé na velikost písmen. Není to již existující atribut.

Shoda podřetězce je ignorována. Porovnávání podřetězců je odpovídající pravidlo použité v dílčím schématu, které určuje chování atributu ve filtru vyhledávání.

ULW *ibm-amqSslPeerName*
Tento atribut LDAP se používá ke kontrole rozlišujícího názvu (DN) certifikátu od správce front typu peer nebo klienta na druhém konci kanálu produktu IBM MQ .

Tento atribut LDAP má jedinou řetězcovou hodnotu s maximálně 1024 bajty, které nejsou citlivé na velikost písmen. Není to již existující.

Shoda podřetězce je ignorována. Porovnávání podřetězců je odpovídající pravidlo použité v dílčím schématu, které určuje chování atributu ve filtru vyhledávání.

ULW *ibm-amqTransactionProgramName*
Tento atribut LDAP uvádí název transakčního programu. Používá se pro připojení LU 6.2 .

Tento atribut má jedinou řetězcovou hodnotu o maximální délce 64 znaků, která nerozlišuje velikost písmen. Není to již existující.

Shoda podřetězce je ignorována. Porovnávání podřetězců je odpovídající pravidlo použité v dílčím schématu, které určuje chování atributu ve filtru vyhledávání.

ULW *ID ibm-amqUser*
Tento atribut LDAP určuje jméno uživatele, které má být použito agentem MCA při pokusu o zahájení zabezpečené relace SNA se vzdáleným agentem MCA.

Tento atribut má jedinou řetězcovou hodnotu přesně 12 znaků, které nejsou citlivé na velikost písmen. Není to již existující atribut.

Shoda podřetězce je ignorována. Porovnávání podřetězců je odpovídající pravidlo použité v dílčím schématu, které určuje chování atributu ve filtru vyhledávání.

ULW *Afinita ibm-amqConnection*
Tento atribut LDAP určuje, zda klientské aplikace, které se připojují vícekrát pomocí stejného názvu správce front, používají stejný kanál klienta.

Tento atribut má jedinou celočíselnou hodnotu. Není to již existující atribut.

ULW *ibm-amqClientChannelWeight*
Tento atribut LDAP určuje váhu ovlivňující definici kanálu připojení klienta, která se má použít.

Váhový atribut kanálu klienta se používá k ovlivnění výběru definic kanálů klienta, je-li k dispozici více než jedna vhodná definice.

Tento atribut má jedinou celočíselnou hodnotu. Není to již existující atribut.

ULW *ibm-amqHeartBeatInterval*
Tento atribut LDAP uvádí přibližný čas mezi toky synchronizačních signálů, které mají být předány z odesílající sběrnice MCA, když v přenosové frontě nejsou žádné zprávy.

Tento atribut má jedinou celočíselnou hodnotu. Není to již existující atribut. Výchozí hodnota je 1. Výchozí hodnota je nastavena v aktuální operaci proměnné prostředí MQSERVER.

ULW *ibm-amqKeepAliveInterval*
Tento atribut LDAP se používá k určení hodnoty časového limitu pro kanál.

Hodnota tohoto atributu se předá do komunikačního zásobníku specifikující časování udržení aktivity pro kanál. Tuto hodnotu můžete použít k určení jiné hodnoty udržení aktivity pro každý kanál.

Tento atribut má jedinou celočíselnou hodnotu. Není to již existující atribut.

ULW *ibm-amqMaximumMessageLength*
Tento atribut LDAP určuje maximální délku zprávy, kterou lze v kanálu přenést.

Výchozí hodnota tohoto atributu je 104857600 tak, jak je za aktuální proměnnou prostředí MQSERVER. Tento atribut má jedinou celočíselnou hodnotu a nejedná se o existující atribut.

ULW *Konverzace ibm-amqSharing*
Tento atribut LDAP uvádí maximální počet konverzací, které sdílejí každou instanci kanálu TCP/IP.

Tento atribut má jedinou celočíselnou hodnotu. Tento atribut není existující atribut.

ULW *Typ ibm-amqTransport*
Tento atribut LDAP určuje typ transportu, který má být použit.

Tento atribut má jedinou celočíselnou hodnotu. Není to již existující atribut.

ULW *ibm-amqIsClientDefault*
Tento logický atribut řeší problém vyhledávání položek, kde atribut ibm-amqQueueManagerName nebyl definován.

Moduly uživatelských procedur před připojením obvykle prohledávají servery LDAP hodnotou atributu ibm-amqQueueManagerName jako vyhledávací kritéria. Takový dotaz by vrátil všechny položky, ve kterých hodnota atributu ibm-amqQueueManagerName odpovídá názvu správce front uvedenému v rámci volání MQCONN/X. Při použití tabulek CCDT (Client Channel Definition CCDT) můžete buď nastavit název správce front na volání MQCONN/X jako prázdný, nebo zadat předponu názvu s hvězdičkou (*). Je-li název správce front prázdný, připojí se klient k výchozímu správci front. Je-li název zadán s hvězdičkou (*) pro správce front, připojí se ke správci front kterýkoli správce front.

Podobně platí, že atribut ibm-amqQueueManagerName v položce může být ponechán nedefinovaný. V tomto případě se očekává, že se klient používající tyto informace koncového bodu může připojit k libovolnému správci front. Položka například obsahuje následující řádky:

```
ibm-amqChannelName = "CHANNEL1"
ibm-amqConnectionName = myhost(1414)
```

V tomto příkladu se klient pokusí o připojení k zadanému správci front spuštěnému v systému `myhost`.

Avšak na serverech LDAP se neprovádí hledání na hodnotě atributu, která nebyla definována. Pokud například položka obsahuje informace o připojení kromě atributu ibm-amqQueueManagerName, výsledky hledání nebudou obsahovat tuto položku. Chcete-li tento problém odstranit, můžete nastavit atribut ibm-amqIsClientDefault. Jedná se o logický atribut a předpokládá se, že má hodnotu FALSE, pokud není definována.

Pro položky, kde parametr ibm-amqQueueManagerName nebyl definován a očekává se, že bude součástí hledání, nastavte atribut ibm-amqIsClientDefault na TRUE. Je-li jako název správce front ve volání MQCONN/X zadán prázdný znak nebo hvězdička (*), uživatelská procedura pro předběžné připojení prohledá server LDAP pro všechny položky, kde hodnota atributu ibm-amqIsClientDefault je nastavena na hodnotu TRUE.

Poznámka: Nenastavujte nebo nadefinujte atribut ibm-amqQueueManagerName , pokud je atribut ibm-amqIsClientDefault nastaven na hodnotu TRUE.

Související odkazy

“ibm-amqQueueManagerName” na stránce 1133
Tento atribut určuje název správce front nebo skupiny správců front, ke které může klientská aplikace klienta IBM MQ požadovat připojení.

ULW *Komprese ibm-amqHeader*
Tento atribut LDAP je seznamem technik komprese dat záhlaví, které jsou podporovány kanálem.

Maximální velikost tohoto atributu je 48 znaků. Není to již existující atribut.

Pro tento atribut můžete zadat pouze jednu hodnotu.

Tento atribut seznamu je určen jako řetězce adresáře pomocí formátu odděleného čárkou. Např. hodnota uvedená pro **ibm-amqHeaderCompression** je `0` , která je mapována na `NONE`. Všechny hodnoty, které překročí maximální povolený limit, budou klientem ignorovány. Například ibm-amqHeaderCompression obsahuje maximálně 2 celá čísla v seznamu.

ULW *Komprese ibm-amqMessage*
Tento atribut LDAP je seznam technik komprese dat zpráv podporovaných kanálem.

Maximální velikost tohoto atributu je 48 znaků. Není to již existující atribut.

Tento atribut nepodporuje více hodnot.

Tento atribut seznamu je určen jako řetězce adresáře pomocí formátu odděleného čárkou. Například, hodnota uvedená pro tento atribut je 1,2,4, která se namapuje na základní pořadí komprese RLE, ZLIBFAST a ZLIBHIGH.

Všechny hodnoty, které překročí maximální povolený limit, budou klientem ignorovány. Například ibm-amqMessageCompression obsahuje maximálně 16 celých čísel v seznamu.

ULW *Data ibm-amqSendExitUserData*
Tento atribut LDAP určuje uživatelská data předávaná uživatelské proceduře pro odeslání zprávy.

Tento atribut LDAP má jedinou řetězcovou hodnotu s maximálně 999 znaky, které nejsou citlivé na velikost písmen. Není to již existující atribut.

Shoda podřetězce je ignorována. Porovnávání podřetězců je odpovídající pravidlo použité v dílčím schématu, které určuje chování atributu ve filtru vyhledávání.

Poznámka: **ibm-amqSendExitName** a **ibm-amqSendExitUserData** je třeba synchronizovat ve dvojicích. Uživatelská data by měla být synchronizována s názvem uživatelské procedury. Je-li tedy jeden zadán, druhý musí být také souměrně specifikován, i když neobsahuje žádná data.

ULW *ibm-amqSendExitName*
Tento atribut LDAP uvádí název ukončovacího programu, který má být spuštěn uživatelskou procedurou odeslání kanálu.

Tento atribut má jedinou řetězcovou hodnotu s maximálně 999 znaky, které nejsou citlivé na velikost písmen. Není to již existující atribut.

Shoda podřetězce je ignorována. Porovnávání podřetězců je odpovídající pravidlo použité v dílčím schématu, které určuje chování atributu ve filtru vyhledávání.

Poznámka: **ibm-amqSendExitName** a **ibm-amqSendExitUserData** musí být synchronizované ve dvojicích. Uživatelská data musí být synchronizována s názvem ukončení. Je-li tedy zadán jeden, musí být druhý také symetricky zadán i v případě, že neobsahuje žádná data.

ULW *Data ibm-amqReceiveExitUserData*
Tento atribut LDAP určuje uživatelská data předávaná uživatelské proceduře pro přijetí zprávy.

Můžete spustit posloupnost uživatelských procedur pro přijetí zprávy. Řetězec uživatelských dat pro řadu uživatelských procedur je oddělen čárkou, mezerami nebo obojím.

Tento atribut má jedinou řetězcovou hodnotu s maximálně 999 znaky, které nejsou citlivé na velikost písmen. Není to již existující atribut.

Shoda podřetězce je ignorována. Porovnávání podřetězců je odpovídající pravidlo použité v dílčím schématu, které určuje chování atributu ve filtru vyhledávání.

Poznámka: **ibm-amqReceiveExitName** a **ibm-amqReceiveExitUserData** musí být synchronizované ve dvojicích. Uživatelská data musí být synchronizována s názvem ukončení. Je-li tedy zadán jeden, musí být druhý také symetricky zadán i v případě, že neobsahuje žádná data.

ULW *ibm-amqReceiveExitName*
Tento atribut LDAP uvádí název uživatelského ukončovacího programu, který má být spuštěn uživatelskou procedurou kanálu pro přijetí zprávy.

Tento atribut je seznam názvů programů, které mají být spuštěny v posloupnosti. Ponechte prázdné, pokud není v platnosti žádná uživatelská procedura příjmu kanálu.

Tento atribut má jedinou řetězcovou hodnotu s maximálně 999 znaky, které nejsou citlivé na velikost písmen. Není to již existující atribut.

Shoda podřetězce je ignorována. Porovnávání podřetězců je odpovídající pravidlo použité v dílčím schématu, které určuje chování atributu ve filtru vyhledávání.

Poznámka: **ibm-amqReceiveExitName** a **ibm-amqReceiveExitUserData** musí být synchronizované ve dvojicích. Uživatelská data musí být synchronizována s názvem ukončení. Je-li tedy zadán, druhý musí být také symetricky zadán, i když neobsahuje žádná data.

z/OS Použití ukázkových programů pro produkt z/OS

Ukázkové procedurální aplikace, které jsou dodávány s produktem IBM MQ for z/OS , demonstrují typická použití rozhraní MQI (Message Queue Interface).

Informace o této úloze

Produkt IBM MQ for z/OS také poskytuje ukázkové uživatelské procedury pro převod dat, popsané v části “Zápis uživatelských procedur pro převod dat” na stránce 948.

Všechny vzorové aplikace jsou dodávány ve zdrojovém tvaru; některé jsou také dodávány ve spustitelném tvaru. Zdrojové moduly obsahují pseudokód, který popisuje logiku programu.

Poznámka: Ačkoli některé z ukázkových aplikací mají základní panelová rozhraní, nemají za cíl demonstrovat, jak navrhnout vzhled a chování vašich aplikací. Další informace o návrhu rozhraní na panelu pro neprogramovatelné svorky naleznete v příručce *SAA Common User Access: Basic Interface Design Guide* (SC26-4583) a jeho dodatku (GG22-9508). Tyto pokyny poskytují pokyny, které vám pomohou navrhnout aplikace, které jsou konzistentní jak v aplikaci, tak i v jiných aplikacích.

Procedura

- Chcete-li zjistit více o ukázkových programech, použijte následující odkazy:
  - “Funkce prokázané v ukázkových aplikacích pro produkt z/OS” na stránce 1138
  - “Příprava a spuštění ukázkových aplikací pro dávkové prostředí v systému z/OS” na stránce 1145
  - “Příprava ukázkových aplikací pro prostředí TSO v systému z/OS” na stránce 1147
  - “Příprava ukázkových aplikací pro prostředí CICS na systému z/OS” na stránce 1149
  - “Příprava ukázkové aplikace pro prostředí produktu IMS v systému z/OS” na stránce 1153
  - “Ukázky vložení v systému z/OS” na stránce 1154
  - “Ukázky získání v systému z/OS” na stránce 1156
  - “Ukázka procházení v systému z/OS” na stránce 1158
  - “Ukázka tiskové zprávy v systému z/OS” na stránce 1160
  - “Ukázka atributů fronty v systému z/OS” na stránce 1164
  - “Ukázka programu Mail Manager v systému z/OS” na stránce 1165
  - “Ukázka Credit Check na z/OS” na stránce 1172
  - “Ukázka obslužné rutiny zpráv v systému z/OS” na stránce 1183
  - “Ukázka Asynchronous Put na systému z/OS” na stránce 1186
  - “The Batch Asynchronous Consumption sample on z/OS” na stránce 1187
  - “Ukázka CICS Asynchronní spotřeba a publikování/odběr v systému z/OS” na stránce 1189
  - “Ukázka Publikování/odběr v systému z/OS” na stránce 1191
  - “Ukázka vlastnosti Message Set a Inquire na objektu z/OS” na stránce 1193

Související úlohy

“Použití ukázkových programů na více platformách” na stránce 1035
Tyto vzorové procedurální programy se dodávají spolu s produktem. Ukázky jsou napsány v jazycích C a COBOL a demonstrují typická použití rozhraní MQI (Message Queue Interface).

z/OS *Funkce prokázané v ukázkových aplikacích pro produkt z/OS*

Tato sekce shrnuje funkce MQI uvedené v každé z ukázkových aplikací, uvádí programovací jazyky, v nichž je každý vzorek napsán, a prostředí, ve kterém je každý vzorek spuštěn.

z/OS *Vložit ukázky do z/OS*

Ukázky Put demonstrují, jak vložit zprávy do fronty pomocí volání MQPUT.

Aplikace používá tato volání MQI:

- MQCONN
- MQOPEN
- MQPUT
- MQCLOSE
- MQDISC

Program se dodává v jazycích COBOL a C a spouští se v dávkovém prostředí a v prostředí CICS . Informace o dávkové aplikaci a Tabulka 168 na stránce 1150 pro aplikaci CICS viz Tabulka 161 na stránce 1145 .

z/OS *Získat ukázky v produktu z/OS*

Ukázky příkazu Get demonstrují způsob získání zpráv z fronty pomocí volání MQGET.

Aplikace používá tato volání MQI:

- MQCONN
- MQOPEN
- MQGET
- MQCLOSE
- MQDISC

Program se dodává v jazycích COBOL a C a spouští se v dávkovém prostředí a v prostředí CICS . Informace o dávkové aplikaci a Tabulka 168 na stránce 1150 pro aplikaci CICS viz Tabulka 161 na stránce 1145 .

z/OS *Procházet ukázku v systému z/OS*

Ukázka Procházet ukazuje, jak použít volbu Procházet k vyhledání zprávy, její vytištění, pak procházet zprávy ve frontě.

Aplikace používá tato volání MQI:

- MQCONN
- MQOPEN
- Příkaz MQGET pro procházení zpráv
- MQCLOSE
- MQDISC

Program je dodáván v jazycích COBOL, assembler, PL/I a C. Aplikace se spustí v prostředí dávkového zpracování. Dávková aplikace viz Tabulka 162 na stránce 1146 .

z/OS *Ukázka tiskové zprávy v systému z/OS*

Ukázka Print Message demonstruje, jak odebrat zprávu z fronty a vytisknout data ve zprávě spolu se všemi poli jeho deskriptoru zpráv. Volitelně může zobrazit všechny vlastnosti zprávy přidružené ke každé zprávě.

Odebráním znaků komentáře ze dvou řádků ve zdrojovém modulu můžete změnit program tak, aby se procházela, spíše než odebírat, zprávy ve frontě. Tento program lze použít k diagnostikování problémů s aplikací, která vkládá zprávy do fronty.

Aplikace používá tato volání MQI:

- MQCONN
- MQOPEN
- MQGET pro odebrání zpráv z fronty (s možností procházení)
- MQCLOSE
- MQDISC
- MQCRTM
- MQDLTMH
- MQINQMP

Program se dodává v jazyce C. Aplikace se spustí v prostředí dávkového zpracování. Dávková aplikace viz Tabulka 163 na stránce 1146 .

z/OS *Ukázka atributů fronty v systému z/OS*

Ukázka Atributy fronty ukazuje, jak se dotázat na a nastavit hodnoty atributů objektu IBM MQ for z/OS .

Aplikace používá tato volání MQI:

- MQOPEN
- MQINQ
- MQSET
- MQCLOSE

Program je dodáván v jazycích COBOL, assembler a C. Aplikace se spustí v prostředí produktu CICS . Informace o aplikaci CICS viz Tabulka 169 na stránce 1151 .

z/OS *Ukázka programu Mail Manager v systému z/OS*

Pokyny týkající se poznámky při použití ukázky Správce pošty.

Ukázka programu Mail Manager demonstruje tyto techniky:

- Použití alias front
- Použití modelové fronty k vytvoření dočasné dynamické fronty
- Použití odpovědí na fronty
- Použití synchronizačních bodů v prostředí CICS a v dávkových prostředích
- Odeslání příkazů do vstupní fronty systému-příkaz
- Testování návratových kódů
- Odesílání zpráv vzdálenému správci front, a to jak pomocí lokální definice vzdálené fronty, tak vložením zpráv přímo do pojmenované fronty ve vzdáleném správci front

Aplikace používá tato volání MQI:

- MQCONN
- MQOPEN
- MQPUT1
- MQGET
- MQINQ
- MQCMIT
- MQCLOSE
- MQDISC

K dispozici jsou tři verze aplikace:

- Aplikace CICS napsaná v jazyce COBOL
- Aplikace TSO napsaná v jazyce COBOL
- Aplikace TSO napsaná v jazyce C

Aplikace TSO používají dávkový adaptér produktu IBM MQ for z/OS a obsahují některé panely ISPF.

See Tabulka 166 na stránce 1148 for the TSO application, and Tabulka 170 na stránce 1151 for the CICS application.

z/OS *Ukázka Credit Check na z/OS*

Tyto informace obsahují body, které je třeba vzít v úvahu při použití ukázky Credit Check.

Ukázka Credit Check je sada programů, které demonstrují tyto techniky:

- Vyvíjení aplikace, která je spuštěna ve více než jednom prostředí
- Použití modelové fronty k vytvoření dočasné dynamické fronty
- Použití identifikátoru korelace
- Nastavení a předání informací o kontextu

- Použití priority zpráv a perzistence
- Spouštění programů pomocí spouštěče
- Použití odpovědí na fronty
- Použití alias front
- Použití fronty nedoručených zpráv
- Použití seznamu názvů
- Testování návratových kódů

Aplikace používá tato volání MQI:

- MQOPEN
- MQPUT
- MQPUT1
- MQGET pro procházení a získávání zpráv, použití voleb čekání a signálů a pro získání specifické zprávy
- MQINQ
- MQSET
- MQCLOSE

Ukázku lze spustit jako samostatnou aplikaci produktu CICS . Chcete-li však demonstrovat, jak navrhnout aplikaci fronty zpráv, která používá prostředky poskytované jak v prostředí CICS , tak IMS , je jeden modul také dodáván jako dávkový program zpracování zpráv IMS .

Programy produktu CICS jsou dodávány v jazycích C a COBOL. Jednotlivý program IMS se dodává v C.

See Tabulka 171 na stránce 1152 for the CICS application, and Tabulka 173 na stránce 1153 for the IMS application.

z/OS *Ukázka obslužné rutiny zpráv v systému z/OS*

Ukázka obslužné rutiny zpráv umožňuje procházet, předávat a odstraňovat zprávy ve frontě.

Aplikace používá tato volání MQI:

- MQCONN
- MQOPEN
- MQINQ
- MQPUT1
- MQCMIT
- MQBACK
- MQGET
- MQCLOSE
- MQDISC

Program je dodáván v jazycích C a COBOL. Aplikace běží pod TSO. Informace o aplikaci TSO vizTabulka 167 na stránce 1149.

z/OS *Distributed queuing exit samples on z/OS*

Tabulka zdrojových programů distribuovaných ukázek uživatelské procedury řazení do fronty.

Názvy zdrojových programů distribuovaných ukázek uživatelské procedury pro řazení do front jsou uvedeny v následující tabulce:

Tabulka 159. Zdroj pro distribuované ukázky uživatelských procedur zařazování do fronty

Název člena	Pro jazyk	Popis	Dodáno v knihovně
CSQ4BAX0	Asembler	Zdrojový program	SSQMY

Tabulka 159. Zdroj pro distribuované ukázky uživatelských procedur zařazování do fronty (pokračování)			
Název člena	**Pro jazyk**	**Popis**	**Dodáno v knihovně**
CSQ4BCX1	C	Zdrojový program	SCSQC37S
CSQ4BCX2	C	Zdrojový program	SCSQC37S
CSQ4BCX4	C	Zdrojový program	SCSQC37S

Poznámka: Zdrojové programy jsou upravovány pomocí CSQXSTUB.

z/OS *Ukázky uživatelské procedury pro převod dat na systému z/OS*

Je poskytnuta kostra pro uživatelskou proceduru pro převod dat a ukázka je dodána s IBM MQ ilustrujícím volání MQXCNVC.

Názvy zdrojových programů pro ukázky uživatelských procedur pro převod dat jsou uvedeny v následující tabulce:

Tabulka 160. Zdroj pro ukázky ukončení převodu dat (pouze jazyk sestavujícího v jazyce)		
Název člena	**Popis**	**Dodáno v knihovně**
CSQ4BAX8	Zdrojový program	SSQMY
CSQ4BAX9	Zdrojový program	SSQMY
CSQ4CAX9	Zdrojový program	SSQMY

Poznámka: Zdrojové programy jsou linovány s CSQASTUB.

Další informace viz “Zápis uživatelských procedur pro převod dat” na stránce 948.

z/OS *Ukázky publikování/odběru v systému z/OS*

Ukázkové programy publikování/odběru demonstrují použití funkcí publikování a odběru v produktu IBM MQ.

K dispozici jsou čtyři programy v jazyku COBOL a dva ukázkové programy programovacího jazyka COBOL, které demonstrují, jak programovat na rozhraní Publikování/odběru produktu IBM MQ .

Aplikace používají tato volání MQI:

- MQCONN
- MQOPEN
- MQPUT
- MQSUB
- MQGET
- MQCLOSE
- MQDISC
- MQCRTM
- MQDLTMH
- MQINQMP

Ukázkové programy Public/Subscribe jsou dodávány v programovacích jazycích C a COBOL. Ukázkové aplikace se spouštějí v dávkovém prostředí. Dávkové aplikace viz Ukázky publikování/odběru .

z/OS ***Konfigurace správce front pro příjem klientských připojení v systému z/OS***

Než budete moci ukázkové aplikace spustit, je třeba nejprve vytvořit správce front. Pak můžete správce front nakonfigurovat tak, aby bezpečně přijímal příchozí požadavky na připojení od aplikací spuštěných v režimu klienta.

Než začnete

Ujistěte se, že správce front již existuje a že byl spuštěn. Určete, zda jsou již záznamy ověřování kanálu povoleny, zadáním příkazu MQSC:

```
DISPLAY QMGR CHLAUTH
```

Důležité: Tato úloha očekává, že jsou povoleny záznamy ověření kanálu. Pokud se jedná o správce front používaný ostatními uživateli a aplikacemi, změna tohoto nastavení bude mít vliv na všechny ostatní uživatele a aplikace. Pokud váš správce front nevyužívá záznamy ověření kanálu, může být krok 4 nahrazen alternativní metodou ověření (například uživatelská procedura zabezpečení), která nastaví MCAUSER na *neprivilegované-ID-uživatele* , které získáte v kroku “1” na stránce 1143.

Musíte vědět, jaký název kanálu očekává vaše aplikace, aby mohla být aplikace povolena pro použití kanálu. Je třeba také vědět, které objekty, například fronty nebo témata, očekává, že aplikace bude používat aplikaci tak, aby je bylo možné používat.

Informace o této úloze

Tato úloha vytvoří neprivilegované ID uživatele, které má být použito pro aplikaci klienta, která se připojuje ke správci front. Přístup pro klientskou aplikaci je udělen pouze k tomu, aby mohl používat kanál, který potřebuje, a frontu, kterou potřebuje k použití tohoto ID uživatele.

Postup

1. Získejte ID uživatele na systému, na kterém je spuštěný správce front.

   Pro tuto úlohu nesmí být toto ID uživatele privilegovaný administrativní uživatel. Toto ID uživatele je oprávnění, pod kterým bude spuštěno připojení klienta ve správci front.
2. Spusťte program listener.
   a) Ujistěte se, že inicializátor kanálu je spuštěn. Pokud tomu tak není, spusťte jej zadáním příkazu **START CHINIT** .
   b) Spusťte program listener zadáním následujícího příkazu:

   ```
   START LISTENER TRPTYPE(TCP) PORT(nnnn)
   ```

   kde *nnnn* je číslo vybraného portu.
3. Používá-li aplikace SYSTEM.DEF.SVRCONN je tento kanál již definován. Pokud vaše aplikace používá jiný kanál, vytvořte jej zadáním příkazu MQSC:

   ```
   DEFINE CHANNEL(' channel-name ') CHLTYPE(SVRCONN) TRPTYPE(TCP) +
   DESCR('Channel for use by sample programs')
   ```

   název-kanálu je název vašeho kanálu.
4. Vytvořte pravidlo pro ověření kanálu, které umožňuje použití kanálu zadáním příkazu MQSC pouze zadáním adresy IP vašeho klienta:

   ```
   SET CHLAUTH(' channel-name ') TYPE(ADDRESSMAP) ADDRESS(' client-machine-IP-address ') +
   MCAUSER(' non-privileged-user-id ')
   ```

   kde:

   název-kanálu je název vašeho kanálu.
   client-machine-IP-address je adresa IP vašeho klientského systému. Je-li ukázková aplikace klienta spuštěna na stejném počítači jako správce front, použijte adresu IP '127.0.0.1', pokud se vaše aplikace chystá připojit pomocí 'localhost'. Pokud se chystáte připojit několik různých klientských počítačů, můžete použít vzor nebo rozsah místo jedné IP adresy. Podrobnosti viz Generické adresy IP .

non-privileged-user-id je ID uživatele, které jste získali v kroku “1” na stránce 1143

5. Používá-li aplikace SYSTEM.DEFAULT.LOCAL.QUEUE, pak je tato fronta již definována. Pokud vaše aplikace používá jinou frontu, vytvořte ji zadáním příkazu MQSC:

```
DEFINE QLOCAL(' queue-name ') DESCR('Queue for use by sample programs')
```

   kde *název-fronty* je název fronty.

6. Udělte přístup pro připojení k správci front a dotazujte jej:
   a) Ujistěte se, že inicializátor kanálu je spuštěn. Pokud ne, spusťte iniciátor kanálu zadáním příkazu START CHINIT.
   b) Spusťte modul listener TCP, například zadejte tento příkaz:

```
START LISTENER TRPTYPE(TCP) PORT(nnnn)
```

   kde *nnnn* je číslo vybraného portu.

7. Je-li vaše aplikace aplikací typu point-to-point, znamená to použití front, udělení přístupu k povolení a odesílání a získání zpráv pomocí vaší fronty pomocí ID uživatele, který má být použit, vydáním příkazů MQSC:

   Zadejte příkazy RACF :

```
RDEFINE MQQUEUE qmgr-name.QUEUE. queue-name UACC(NONE)

PERMIT qmgr-name.QUEUE. queue-name CLASS(MQQUEUE) ID(non-privileged-user-id) ACCESS(UPDATE)
```

   kde:

   qmgr-name je název vašeho správce front
   queue-name je název vaší fronty.
   non-privileged-user-id je ID uživatele, které jste získali v kroku “1” na stránce 1143

8. Je-li vaše aplikace aplikací typu publikování-odběr, která využívá témata, udělte přístup k povolení publikování a přihlašování pomocí vašeho tématu pomocí ID uživatele, který má být použit, zadáním následujících příkazů produktu RACF :

```
RDEFINE MQTOPIC qmgr-name.PUBLISH.SYSTEM.BASE.TOPIC UACC(NONE)

PERMIT qmgr-name.PUBLISH.SYSTEM.BASE.TOPIC CLASS(MQTOPIC) ID(non-privileged-user-id)
ACCESS(UPDATE)

RDEFINE MQTOPIC qmgr-name.SUBSCRIBE.SYSTEM.BASE.TOPIC UACC(NONE)

PERMIT qmgr-name.SUBSCRIBE.SYSTEM.BASE.TOPIC CLASS(MQTOPIC) ID(non-privileged-user-id)
ACCESS(UPDATE)
```

   kde:

   qmgr-name je název vašeho správce front
   non-privileged-user-id je ID uživatele, které jste získali v kroku “1” na stránce 1143
   Tato akce poskytne uživateli *neprivilegované-ID-uživatele* přístup k libovolnému tématu ve stromu témat, případně můžete definovat objekt tématu pomocí produktu **DEFINE TOPIC** a udělit přístup pouze k části stromu témat odkazovaného daným objektem tématu. Další informace naleznete v tématu Řízení přístupu uživatelů k tématům.

Jak pokračovat dále

Klientská aplikace se nyní může připojit ke správci front a odesílat nebo přijímat zprávy s použitím fronty.

Související informace

SET CHLAUTH
Definovat kanál
DEFINOVAT QLOCAL

SET AUTHREC

z/OS Oprávnění pro práci s objekty IBM MQ v systému z/OS

z/OS *Příprava a spuštění ukázkových aplikací pro dávkové prostředí v systému z/OS*

Chcete-li připravit ukázkovou aplikaci, která se spustí v prostředí dávkového zpracování, proveďte stejné kroky, jako byste při sestavování aplikace dávky IBM MQ for z/OS .

Tyto kroky jsou uvedeny v tématu “Sestavení dávkových aplikací produktu z/OS” na stránce 1001.

Případně, pokud dodáváme spustitelnou formu ukázky, můžete ji spustit z zaváděcí knihovny thlqual.SCSQLOAD .

Poznámka: Verze jazyka v asembleru pro ukázkovou aplikaci Procházet používá řídicí bloky dat (DCB), takže ji musíte propojit-upravit jej pomocí produktu RMODE(24).

Členy knihovny, které se mají použít, jsou vypsány v Tabulka 161 na stránce 1145, Tabulka 162 na stránce 1146, Tabulka 163 na stránce 1146a Tabulka 164 na stránce 1147.

Musíte upravit kód JCL spuštění dodaný pro ukázky, které chcete použít (viz Tabulka 161 na stránce 1145, Tabulka 162 na stránce 1146, Tabulka 163 na stránce 1146a Tabulka 164 na stránce 1147).

Příkaz PARM v dodaném JCL obsahuje řadu parametrů, které je třeba upravit. Chcete-li spustit ukázkové programy v jazyku C, oddělte parametry mezerami; chcete-li spustit ukázkové programy v asembleru, COBOL a PL/I, oddělte je čárkami. Je-li například název správce front CSQ1 a chcete-li spustit aplikaci s frontou s názvem LOCALQ1, v jazyce JCL jazyka COBOL, PL/I a v souboru JCL sestavujícího jazyka, měl by příkaz PARM vypadat takto:

```
PARM=(CSQ1,LOCALQ1)
```

V jazyce JCL jazyka C by váš příkaz PARM měl vypadat takto:

```
PARM=('CSQ1 LOCALQ1')
```

Nyní jste připraveni odeslat úlohy.

z/OS *Názvy ukázkových dávkových aplikací v systému z/OS*

Souhrn programů, které jsou dodávány pro ukázkové dávkové aplikace.

Dávkové aplikační programy jsou shrnuty v následujících tabulkách:

- Tabulka 161 na stránce 1145 Načíst a získat ukázky
- Tabulka 162 na stránce 1146 Procházet ukázky
- Tabulka 163 na stránce 1146 Ukázka tiskové zprávy
- Ukázky publikování/odběru Tabulka 164 na stránce 1147
- Tabulka 165 na stránce 1147 Ostatní ukázky

Tabulka 161. Dávkové vkládání a získávání ukázek

Název člena	Pro jazyk	Popis	Zdrojový soubor dodaný v knihovně	Spustitelný soubor dodaný v knihovně
CSQ4BCJ1	C	Získat zdrojový program	SCSQC37S	SCQLOAD
CSQ4BCK1	C	Vložit zdrojový program	SCSQC37S	SCQLOAD

Tabulka 161. Dávkové vkládání a získávání ukázek (pokračování)

Název člena	Pro jazyk	Popis	Zdrojový soubor dodaný v knihovně	Spustitelný soubor dodaný v knihovně
CSQ4BCJR	C	Ukázka spuštění JCL pro CSQ4BCJ1 a CSQBCK1	SCSQPROC	Není
CSQ4BVJ1	COBOL	Získat zdrojový program	SCSQCOBS	SCQLOAD
CSQ4BVK1	COBOL	Vložit zdrojový program	SCSQCOBS	SCQLOAD
CSQ4BVJR	COBOL	Ukázka spuštění JCL pro CSQBVJ1 a CSQBVK1	SCSQPROC	Není

Tabulka 162. Ukázka dávkového procházení

Název člena	Pro jazyk	Popis	Zdrojový soubor dodaný v knihovně	Spustitelný soubor dodaný v knihovně
CSQ4BVA1	COBOL	Zdrojový program	SCSQCOBS	SCQLOAD
CSQ4BVAR	COBOL	Ukázka spuštění JCL pro CSQ4BVA1	SCSQPROC	Není
CSQ4BAA1	Asembler	Zdrojový program	SSQMY	SCQLOAD
CSQ4BAAR	Asembler	Ukázka kódu JCL spuštění pro CSQ4BAA1	SCSQPROC	Není
CSQ4BCA1	C	Zdrojový program	SCSQC37S	SCQLOAD
CSQ4BCAR	C	Ukázka kódu JCL spuštění pro CSQ4BCA1	SCSQPROC	Není
CSQ4BPA1	PL/I	Zdrojový program	SCSQPLAS	SCQLOAD
CSQ4BPAR	PL/I	Ukázka spuštění JCL pro CSQ4BPA1	SCSQPROC	Není

Tabulka 163. Ukázka Dávková tisková zpráva (pouze jazyk C)

Název člena	Popis	Zdrojový soubor dodaný v knihovně	Spustitelný soubor dodaný v knihovně
CSQ4BCG1	Zdrojový program	SCSQC37S	SCQLOAD
CSQ4BCGR	Ukázka kódu JCL spuštění pro CSQ4BCG1	SCSQPROC	Není
CSQ4BCL1	Procházet zdrojový program	SCSQC37S	SCQLOAD
CSQ4BCLR	Ukázka spuštění JCL pro CSQ4BCL1	SCSQPROC	Není

Tabulka 164. Ukázky publikování/odběru

Název člena	Pro jazyk	Popis	Zdrojový soubor dodaný v knihovně	JCL v SCSQPROC	Spustitelný soubor dodaný v knihovně
CSQ4BCP1	C	Publikovat do zdrojového programu tématu	SCSQC37S	CSQ4BCPP	SCQLOAD
CSQ4BCP2	C	Přihlásit se k odběru tématu a získání zdrojového programu zpráv	SCSQC37S	CSQ4BCPS	SCQLOAD
CSQ4BCP3	C	Přihlaste se k odběru tématu pomocí cíle poskytnutého uživatelem a získejte zdrojový program zpráv	SCSQC37S	CSQ4BCPD	SCQLOAD
CSQ4BCP4	C	Přihlásit se k odběru tématu pomocí rozšířených voleb a získání zdrojového programu zpráv	SCSQC37S	CSQ4BCPE	SCQLOAD
CSQ4BVP1	COBOL	Publikovat do zdrojového programu tématu	SCSQCOBS	CSQ4BVPP	SCQLOAD
CSQ4BVP2	COBOL	Přihlásit se k odběru tématu a získání zdrojového programu zpráv	SCSQCOBS	CSQ4BVPS	SCQLOAD

Tabulka 165. Ostatní vzorky

Název člena	Pro jazyk	Popis	Zdrojový soubor dodaný v knihovně	JCL v SCSQPROC	Spustitelný soubor dodaný v knihovně
CSQ4BCS1	C	Zdrojový program asynchronní spotřeby	SCSQC37S	CSQ4BCSC	SCQLOAD
CSQ4BCS2	C	Zdrojový program Asynchronous Put a Check.	SCSQC37S	CSQ4BCSP	SCQLOAD
CSQ4BCM1	C	Zjistit zdrojový program vlastností zprávy	SCSQC37S	CSQ4BCMP	SCQLOAD
CSQ4BCM2	C	Nastavit zdrojový program vlastností zprávy	SCSQC37S	CSQ4BCMP	SCQLOAD

z/OS *Příprava ukázkových aplikací pro prostředí TSO v systému z/OS*

Chcete-li připravit ukázkovou aplikaci, která je spuštěna v prostředí TSO, proveďte stejné kroky, jako byste při sestavování aplikace produktu IBM MQ for z/OS vytvářeli.

Tyto kroky jsou uvedeny v tématu “Sestavení dávkových aplikací produktu z/OS” na stránce 1001. Členy knihovny, které se mají použít, jsou vypsány v Tabulka 166 na stránce 1148.

Případně, pokud dodáváme spustitelnou formu ukázky, můžete ji spustit z zaváděcí knihovny thlqual.SCSQLOAD .

V případě ukázkové aplikace správce pošty zkontrolujte, zda jsou ve vašem systému k dispozici fronty, které používá. Jsou definovány ve členu **thlqual**.SCSQPROC (CSQ4CVD). Chcete-li zajistit, že tyto fronty budou vždy k dispozici, můžete tyto členy přidat do vstupní datové sady inicializace CSQINP2 nebo použít program CSQUTIL k načtení těchto definic front.

z/OS *Názvy ukázkových aplikací TSO v systému z/OS*

Informace o názvech programů, které jsou dodávány pro každou z ukázkových aplikací TSO, a knihovny, kde je zdrojový kód, JCL, a pouze pro ukázku popisovače zpráv jsou umístěny spustitelné soubory.

Aplikační programy TSO jsou shrnuty v následujících tabulkách:

- Tabulka 166 na stránce 1148 Ukázka správce pošty
- Tabulka 167 na stránce 1149 Ukázka obslužné rutiny zpráv

Tyto ukázky používají panely ISPF. Musíte proto zahrnout stub ISPF, ISPLINK, když linkování-upravit programy.

Tabulka 166. Ukázka TSO Mail Manager

Název člena	**Pro jazyk**	**Popis**	**Zdrojový soubor dodaný v knihovně**
CSQ4CVD	Nezávislý	Definice objektů produktu IBM MQ for z/OS	SCSQPROC
CSQ40	Nezávislý	Zprávy ISPF	SCSQMSGE
CSQ4RVD1	COBOL	CLIST pro zahájení CSQ4TVD1	SCSQCLST
CSQ4TVD1	COBOL	Zdrojový program pro program nabídky	SCSQCOBS
CSQ4TVD2	COBOL	Zdrojový program pro program Get Mail	SCSQCOBS
CSQ4TVD4	COBOL	Zdrojový program pro odeslání poštovního programu	SCSQCOBS
CSQ4TVD5	COBOL	Zdrojový program pro program přezdívek	SCSQCOBS
CSQ4VDP1-6	COBOL	Definice panelů	SCSQPNLA
CSQ4VD0	COBOL	definice dat	SCSQCOBC
CSQ4VD1	COBOL	definice dat	SCSQCOBC
CSQ4VD2	COBOL	definice dat	SCSQCOBC
CSQ4VD4	COBOL	definice dat	SCSQCOBC
CSQ4RCD1	C	CLIST pro zahájení CSQ4TCD1	SCSQCLST
CSQ4TCD1	C	Zdrojový program pro program nabídky	SCSQC37S

Tabulka 166. Ukázka TSO Mail Manager (pokračování)			
Název člena	**Pro jazyk**	**Popis**	**Zdrojový soubor dodaný v knihovně**
CSQ4TCD2	C	Zdrojový program pro program Get Mail	SCSQC37S
CSQ4TCD4	C	Zdrojový program pro odeslání poštovního programu	SCSQC37S
CSQ4TCD5	C	Zdrojový program pro program přezdívek	SCSQC37S
CSQ4CDP1-6	C	Definice panelů	SCSQPNLA
CSQ4TC0	C	Zahrnout soubor	SCSQC370

Tabulka 167. Ukázka popisovače zpráv TSO				
Název člena	**Pro jazyk**	**Popis**	**Zdrojový soubor dodaný v knihovně**	**Spustitelný soubor dodaný v knihovně**
CSQ4TCH0	C	definice dat	SCSQC370	Není
CSQ4TCH1	C	Zdrojový program	SCSQC37S	SCQLOAD
CSQ4TCH2	C	Zdrojový program	SCSQC37S	SCQLOAD
CSQ4TCH3	C	Zdrojový program	SCSQC37S	SCQLOAD
CSQ4RCH1	C a COBOL	CLIST pro zahájení CSQ4TCH1 nebo CSQ4TVH1	SCSQCLST	Není
CSQ4CHP1	C a COBOL	Definice panelu	SCSQPNLA	Není
CSQ4CHP2	C a COBOL	Definice panelu	SCSQPNLA	Není
CSQ4CHP3	C a COBOL	Definice panelu	SCSQPNLA	Není
CSQ4CHP9	C a COBOL	Definice panelu	SCSQPNLA	Není
CSQ4TVH0	COBOL	definice dat	SCSQCOBC	Není
CSQ4TVH1	COBOL	Zdrojový program	SCSQCOBS	SCQLOAD
CSQ4TVH2	COBOL	Zdrojový program	SCSQCOBS	SCQLOAD
CSQ4TVH3	COBOL	Zdrojový program	SCSQCOBS	SCQLOAD

z/OS *Příprava ukázkových aplikací pro prostředí CICS na systému z/OS*

Před spuštěním ukázkových programů produktu CICS se přihlaste k produktu CICS pomocí příkazu LOGMODE 32702. Důvodem je to, že ukázkové programy byly napsány tak, aby používaly obrazovku režimu 3270 typu 2.

Chcete-li připravit ukázkovou aplikaci, která se spustí v prostředí produktu CICS , postupujte takto:

1. Vytvořte mapu symbolického popisu a fyzickou mapu obrazovky pro ukázku tak, že sestavíte zdroj definice obrazovky BMS (dodaný v knihovně **thlqual**.SCSQMAPS, kde **thlqual** je vysokoúrovňový kvalifikátor použitý při instalaci). Když pojmenujete mapy, použijte název zdroje definice obrazovky BMS (není k dispozici pro ukázkové programy Put and Get), ale vynechte poslední znak názvu.

2. Proveďte stejné kroky, jaké byste měli při sestavování aplikace produktu CICS IBM MQ for z/OS . Tyto kroky jsou uvedeny v tématu “Sestavování aplikací produktu CICS v produktu z/OS” na stránce 1004. Členy knihovny, které se mají použít, jsou vypsány v Tabulka 168 na stránce 1150, Tabulka 169 na stránce 1151, Tabulka 170 na stránce 1151a Tabulka 171 na stránce 1152.

   Případně, pokud dodáváme spustitelnou formu ukázky, můžete ji spustit z knihovny načtení thlqual.SCSQCICS .

3. Pomocí aktualizace datové sady definic systému CICS (CSD) identifikujte sadu map, programy a transakci na CICS . Definice, které vyžadujete, jsou ve členu **thlqual**.SCSQPROC (CSQ4S100). Pokyny k tomuto postupu najdete v části *Adaptér CICS-IBM MQ* v dokumentaci produktu CICS Transaction Server 4.1 for z/OS na adrese: CICS Transaction Server 4.1 for z/OS, The CICS-IBM MQ adapter.

   Poznámka: Pro ukázkovou aplikaci Kontrola kreditu se v této fázi zobrazí chybová zpráva, pokud jste již nevytvořili datovou sadu VSAM, kterou ukázka používá.

4. Pro ukázkové aplikace kontroly kreditu a správce pošty se ujistěte, že fronty, které používají, jsou ve vašem systému k dispozici. Pro ukázku Credit Check, jsou definovány ve členu **thlqual**.SCSQPROC (CSQ4CVB) pro COBOL a **thlqual**.SCSQPROC (CSQ4CCB) pro C. Pro ukázku programu Mail Manager jsou definovány ve členu **thlqual**.SCSQPROC (CSQ4CVD). Chcete-li zajistit, že tyto fronty budou vždy k dispozici, můžete tyto členy přidat do vstupní datové sady inicializace CSQINP2 nebo použít program CSQUTIL k načtení těchto definic front.

   Pro ukázkovou aplikaci Atributy fronty můžete použít jednu nebo více front, které jsou dodávány pro další ukázkové aplikace. Alternativně můžete použít vlastní fronty. Avšak, ve formě, která je dodána, tato ukázka pracuje pouze s frontami, které mají znaky CSQ4SAMP v prvních osmi bajtech jejich názvu.

z/OS *Názvy ukázkových aplikací produktu CICS v systému z/OS*

Toto téma obsahuje souhrn programů dodávaných s ukázkovými aplikacemi produktu CICS .

Aplikační programy produktu CICS jsou shrnuty v následujících tabulkách:

- Tabulka 168 na stránce 1150 Načíst a získat ukázky
- Tabulka 169 na stránce 1151 Ukázka atributů fronty
- Ukázka programu Tabulka 170 na stránce 1151 Mail Manager (pouze COBOL)
- Tabulka 171 na stránce 1152 Ukázka Credit Check
- Tabulka 172 na stránce 1153 Ukázky asynchronní spotřeby a publikování/odběru

Tabulka 168. CICS Načíst a získat ukázky

Název člena	Pro jazyk	Popis	Zdrojový soubor dodaný v knihovně	Spustitelný soubor dodaný v knihovně
CSQ4CCK1	C	Vložit zdrojový program	SCSQC37S	SCSQCICS
CSQ4CCJ1	C	Získat zdrojový program	SCSQC37S	SCSQCICS
CSQ4CVJ1	COBOL	Získat zdrojový program	SCSQCOBS	SCSQCICS
CSQ4CVK1	COBOL	Vložit zdrojový program	SCSQCOBS	SCSQCICS
CSQ4S100	Nezávislý	Datová sada definice systému CICS	SCSQPROC	Není

Tabulka 169. CICS Ukázka atributů fronty

Název člena	Pro jazyk	Popis	Zdrojový soubor dodaný v knihovně	Spustitelný soubor dodaný v knihovně
CSQ4CVC1	COBOL	Zdrojový program	SCSQCOBS	SCSQCICS
CSQ4VMSG	COBOL	Definice zprávy	SCSQCOBC	Není
CSQ4VCMS	COBOL	Definice obrazovky BMS	SCQMAPS	SCSQCICS (s názvem CSQ4ACM)
CSQ4CAC1	Asembler	Zdrojový program	SSQMY	SCSQCICS
CSQ4AMSG	Asembler	Definice zprávy	SCSQMACS	Není
CSQ4ACMS	Asembler	Definice obrazovky BMS	SCQMAPS	SCSQCICS (s názvem CSQ4ACM)
CSQ4CCC1	C	Zdrojový program	SCSQC37S	SCSQCICS
CSQ4CMSG	C	Definice zprávy	SCSQC370	Není
CSQ4CCMS	C	Definice obrazovky BMS	SCQMAPS	SCSQCICS (s názvem CSQ4ACM)
CSQ4S100	Nezávislý	Datová sada definice systému CICS	SCSQPROC	Není

Tabulka 170. Ukázka programu CICS Mail Manager (pouze COBOL)

Název člena	Popis	Zdrojový soubor dodaný v knihovně
CSQ4CVD	Definice objektů produktu IBM MQ for z/OS	SCSQPROC
CSQ4CVD1	Zdroj pro program menu	SCSQCOBS
CSQ4CVD2	Zdroj pro získání poštovního programu	SCSQCOBS
CSQ4CVD3	Zdroj pro program zobrazení zpráv	SCSQCOBS
CSQ4CVD4	Zdroj pro program pro odeslání pošty	SCSQCOBS
CSQ4CVD5	Zdroj pro program přezdívek	SCSQCOBS
CSQ4VDMS	Zdroj definice obrazovky BMS	SCQMAPS
CSQ4S100	Datová sada definice systému CICS	SCSQPROC
CSQ4VD0	definice dat	SCSQCOBC
CSQ4VD3	definice dat	SCSQCOBC
CSQ4VD4	definice dat	SCSQCOBC

Tabulka 171. CICS Ukázka Credit Check

Název člena	Pro jazyk	Popis	Zdrojový soubor dodaný v knihovně
CSQ4CVB	Nezávislý	Definice objektů produktu IBM MQ	SCSQPROC
CSQ4CCB	Nezávislý	Definice objektů produktu IBM MQ	SCSQPROC
CSQ4CVB1	COBOL	Zdroj pro program uživatelského rozhraní	SCSQCOBS
CSQ4CVB2	COBOL	Zdroj pro správce kreditních aplikací	SCSQCOBS
CSQ4CVB3	COBOL	Zdroj pro program check-account	SCSQCOBS
CSQ4CVB4	COBOL	Zdroj pro distribuční program	SCSQCOBS
CSQ4CVB5	COBOL	Zdroj pro agenturu-dotaz program	SCSQCOBS
CSQ4CCB1	C	Zdroj pro program uživatelského rozhraní	SCSQC37S
CSQ4CCB2	C	Zdroj pro správce kreditních aplikací	SCSQC37S
CSQ4CCB3	C	Zdroj pro program check-account	SCSQC37S
CSQ4CCB4	C	Zdroj pro distribuční program	SCSQC37S
CSQ4CCB5	C	Zdroj pro agenturu-dotaz program	SCSQC37S
CSQ4CB0	C	Zahrnout soubor	SCSQC370
CSQ4CBMS	C	Zdroj definice obrazovky BMS	SCQMAPS
CSQ4VBMS	COBOL	Zdroj definice obrazovky BMS	SCQMAPS
CSQ4VB0	COBOL	definice dat	SCSQCOBC
CSQ4VB1	COBOL	definice dat	SCSQCOBC
CSQ4VB2	COBOL	definice dat	SCSQCOBC
CSQ4VB3	COBOL	definice dat	SCSQCOBC
CSQ4VB4	COBOL	definice dat	SCSQCOBC
CSQ4VB5	COBOL	definice dat	SCSQCOBC
CSQ4VB6	COBOL	definice dat	SCSQCOBC
CSQ4VB7	COBOL	definice dat	SCSQCOBC
CSQ4VB8	COBOL	definice dat	SCSQCOBC
CSQ4BAQ	Nezávislý	Zdroj pro datovou sadu VSAM	SCSQPROC
CSQ4FILE	Nezávislý	JCL pro sestavení datové sady VSAM použité CSQ4CVB3	SCSQPROC
CSQ4S100	Nezávislý	Datová sada definice systému CICS	SCSQPROC

Tabulka 172. CICS Ukázky asynchronní spotřeby a publikování/odběru		
Název člena	**Popis**	**Zdrojový soubor dodaný v knihovně**
CSQ4CVCN	Zdroj pro program Simple Message Consumption	SCSQCOBS
CSQ4CVCT	Zdroj programu pro řízení spotřeby zpráv	SCSQCOBS
CSQ4CVEV	Zdroj pro program obslužné rutiny událostí	SCSQCOBS
CSQ4CVPT	Zdroj pro program Message Put Client	SCSQCOBS
CSQ4CVRG	Zdroj pro program registračního klienta	SCSQCOBS
CSQ4S100	Datová sada definice systému CICS	SCSQPROC

z/OS *Příprava ukázkové aplikace pro prostředí produktu IMS v systému z/OS*

Část ukázkové aplikace Credit Check (Credit Check) může být spuštěna v prostředí produktu IMS .

Chcete-li připravit tuto část aplikace tak, aby se spouštěl s ukázkou produktu CICS , nejprve proveďte kroky popsané v tématu “Příprava ukázkových aplikací pro prostředí CICS na systému z/OS” na stránce 1149.

Poté proveďte následující kroky:

1. Proveďte stejné kroky, jaké byste měli při sestavování aplikace produktu IMS IBM MQ for z/OS . Tyto kroky jsou uvedeny v tématu “Sestavování aplikací IMS (BMP nebo MPP)” na stránce 1005. Členy knihovny, které se mají použít, jsou vypsány v Tabulka 173 na stránce 1153.
2. Identifikujte aplikační program a databázi na IMS. Ukázky jsou poskytovány s příkazy PSBGEN, DBDGEN, ACB, IMSGEN a IMSDALOC, aby bylo možné toto povolit.
3. Načtěte databázi CSQ4CA na základě krejením a spuštěním ukázkového JCL poskytnutého pro tento účel (CSQ4ILDB). Tento skript JCL načte databázi s daty ze souboru CSQ4BAQ. Aktualizujte řídicí oblast produktu IMS pomocí příkazu definice dat pro databázi CSQ4CA.
4. Spusťte program checking-account jako program dávkového zpracování zpráv (BMP) tím, že přizpůsobíte a spustíte ukázkový JCL poskytnutý pro tento účel. Tento kód JCL spouští dávkový program BMP orientovaný na dávky. Chcete-li program spustit jako program BMP orientovaný na zprávy, odeberte z řádku v souboru JCL, který obsahuje příkaz IN=, znaky komentáře.

z/OS *Názvy ukázkové aplikace IMS v systému z/OS*

Tyto informace poskytují tabulku se seznamem zdrojů a JCLs, které jsou dodány pro ukázkovou aplikaci IMS .

Tabulka 173. Zdroj a kód JCL pro vzorek Kreditní kontroly IMS (pouze jazyk C)		
Název člena	**Popis**	**Dodáno v knihovně**
CSQ4CVB	Definice objektů produktu IBM MQ	SCSQPROC
CSQ4ICB3	Zdroj pro program check-account	SCSQC37S
CSQ4ICBL	Zdroj pro načtení databáze checking-account	SCSQC37S

Tabulka 173. Zdroj a kód JCL pro vzorek Kreditní kontroly IMS (pouze jazyk C) (pokračování)		
Název člena	**Popis**	**Dodáno v knihovně**
CSQ4CBI	definice dat	SCSQC370
CSQ4PSBL	PSBGEN JCL pro databázový program-load program	SCSQPROC
CSQ4PSB3	Kód PSBGEN JCL pro program checking-account	SCSQPROC
CSQ4DBDS	DBDGEN JCL pro databázi CSQ4CA	SCSQPROC
CSQ4GIMS	Definice maker produktu IMSGEN pro CSQ4IVB3 a CSQ4CA	SCSQPROC
CSQ4ACBG	Definice ovládacího prvku řízení aplikace (ACB) pro CSQ4IVB3	SCSQPROC
CSQ4BAQ	Zdroj pro databázi	SCSQPROC
CSQ4ILDB	Ukázka spuštění skriptu JCL pro úlohu načtení databáze	SCSQPROC
CSQ4ICBR	Ukázka spuštění skriptu JCL pro program checking-account	SCSQPROC
CSQ4DYNA	IMSDALOC macro definice pro databázi	SCSQPROC

z/OS *Ukázky vložení v systému z/OS*

Ukázkové programy vkládání vloží zprávy do fronty pomocí volání MQPUT.

Zdrojové programy jsou dodávány v jazycích C a COBOL v dávkách a v prostředích CICS (viz Tabulka 161 na stránce 1145 a Tabulka 168 na stránce 1150).

Návrh vzorku Put

Průtok logikou programu je:

1. Připojte se ke správci front pomocí volání MQCONN. Pokud se toto volání nezdaří, vytiskněte kódy dokončení a příčiny a zastavte zpracování.

   Poznámka: Spouštíte-li ukázku v prostředí produktu CICS , nemusíte zadávat volání MQCONN; pokud ano, vrátí hodnotu DEF_HCONN. Pro následující volání MQI můžete použít manipulátor připojení MQHC_DEF_HCONN pro volání MQI.
2. Otevřete frontu pomocí volání MQOPEN s volbou MQOO_OUTPUT. Na vstupu do tohoto volání program používá popisovač připojení, který je vrácen v kroku “1” na stránce 1156. Pro strukturu deskriptoru objektu (MQOD) používá výchozí hodnoty pro všechna pole kromě pole názvu fronty, která se předává programu jako parametr programu. Pokud se volání MQOPEN nezdaří, vytiskněte kódy dokončení a příčiny a zastavte zpracování.
3. Vytvořte smyčku v rámci programu, který vydá volání MQPUT, dokud se do fronty nevloží požadovaný počet zpráv. Pokud volání MQPUT selže, je smyčka opuštěna předčasně, nedojde k pokusu o další volání MQPUT a jsou vráceny kódy dokončení a příčiny.
4. Zavřete frontu s použitím volání MQCLOSE s manipulátorem objektu vráceným v kroku “2” na stránce 1156. Pokud toto volání selže, vytiskněte kódy dokončení a příčiny.
5. Odpojte se od správce front s použitím volání MQDISC s manipulátorem připojení vráceným v kroku “1” na stránce 1156. Pokud toto volání selže, vytiskněte kódy dokončení a příčiny.

   Poznámka: Pokud spouštíte ukázku v prostředí produktu CICS , není třeba volat volání MQDISC.

z/OS *Ukázky Put pro dávkové prostředí na z/OS*

Toto téma použijte při zvažování vkládání vzorků pro dávkové prostředí.

Chcete-li ukázky spustit, upravte a spusťte ukázku JCL, jak je popsáno v tématu “Příprava a spuštění ukázkových aplikací pro dávkové prostředí v systému z/OS” na stránce 1145.

Programy používají následující parametry v EXEC PARM, oddělené mezerami v C a čárek v COBOLu:

1. Název správce front (4 znaky)
2. Název cílové fronty (48 znaků)
3. Počet zpráv (až 4 číslice)
4. Výplň, která se má zapsat ve zprávě (1 znak)
5. Počet znaků, které se mají zapsat ve zprávě (maximálně 4 číslice)
6. Perzistence zprávy (1 znak: P pro trvalé nebo N pro přechodnou dobu)

Zadáte-li některý z těchto parametrů nesprávně, obdržíte příslušné chybové zprávy.

Všechny zprávy z ukázek jsou zapsány do datové sady SYSPRINT.

Poznámky k použití

- Chcete-li uchovat ukázky jednoduché, existují menší funkční rozdíly mezi jazykovými verzemi. Tyto rozdíly jsou však minimalizovány, pokud použijete rozvržení parametrů zobrazených v ukázce spuštění JCL, CSQ4BCJRa CSQ4BVJR. Žádný z rozdílů se netýká rozhraní MQI.
- CSQ4BCK1 vám umožňuje zadat více než čtyři číslice pro počet odeslaných zpráv a délku zpráv.
- Pro dvě numerická pole zadejte libovolnou číslici v rozsahu od 1 do 9999. Hodnota, kterou zadáte, by měla být kladné číslo. Chcete-li například vložit jednu zprávu, můžete jako hodnotu zadat 1, 01, 001 nebo 0001. Zadáte-li nečíselné nebo záporné hodnoty, může dojít k chybě. Zadáte-li například hodnotu -1, program v jazyce COBOL odešle jednobajtovou zprávu, ale program C obdrží chybu.
- Pro oba programy, CSQ4BCK1 a CSQ4BVK1, musíte zadat P v parametru perzistence, + + PER + +, pokud chcete, aby zpráva byla trvalá. Pokud tomu tak neuděláte, bude zpráva netrvalá.

z/OS *Ukázky Put pro prostředí produktu CICS v systému z/OS*

Toto téma použijte při zvažování vkládání ukázek do prostředí produktu CICS .

Transakce mají následující parametry oddělené čárkami:

1. Počet zpráv (až 4 číslice)
2. Výplň, která se má zapsat ve zprávě (1 znak)
3. Počet znaků, které se mají zapsat ve zprávě (maximálně 4 číslice)
4. Perzistence zprávy (1 znak: P pro trvalé nebo N pro přechodnou dobu)
5. Název cílové fronty (48 znaků)

Zadáte-li některý z těchto parametrů nesprávně, obdržíte příslušné chybové zprávy.

Pro ukázku jazyka COBOL vyvolejte ukázku Vložení v prostředí produktu CICS zadáním následujícího příkazu:

```
MVPT,9999,*,9999,P,QUEUE.NAME
```

V případě ukázky C vyvolejte ukázku Vložit v prostředí produktu CICS zadáním:

```
MCPT,9999,*,9999,P,QUEUE.NAME
```

Všechny zprávy z ukázek se zobrazí na obrazovce.

Poznámky k použití

- Chcete-li uchovat ukázky jednoduché, existují menší funkční rozdíly mezi jazykovými verzemi. Žádný z rozdílů se netýká rozhraní MQI.
- Pokud zadáte název fronty delší než 48 znaků, jeho délka je oříznuta na maximálně 48 znaků, ale nevrátí se žádná chybová zpráva.
- Před zadáním transakce stiskněte klávesu CLEAR.
- Pro dvě numerická pole zadejte jakékoli číslo v rozsahu od 1 do 9999. Hodnota, kterou zadáte, by měla být kladné číslo. Chcete-li například vložit jednu zprávu, můžete zadat hodnotu 1, 01, 001 nebo 0001. Zadáte-li nečíselné nebo záporné hodnoty, může dojít k chybě. Zadáte-li například hodnotu -1, program v jazyce COBOL odešle jednobajtovou zprávu a program v jazyce C bude ukončen s chybou funkcí malloc ().
- Pro oba programy, CSQ4CCK1 a CSQ4CVK1zadejte P v parametru perzistence, pokud chcete, aby zpráva byla trvalá. Pro netrvalé zprávy zadejte do parametru perzistence hodnotu N. Zadáte-li jakoukoli jinou hodnotu, obdržíte chybovou zprávu.
- Zprávy jsou vloženy do synchronizačního bodu, protože výchozí hodnoty se používají pro všechny parametry kromě těch, které jsou nastaveny během vyvolání programu.

z/OS *Ukázky získání v systému z/OS*

Vzorové programy typu Get získají zprávy z fronty pomocí volání MQGET.

Zdrojové programy jsou dodávány v jazycích C a COBOL v dávkách a v prostředích CICS (viz Tabulka 161 na stránce 1145 a Tabulka 168 na stránce 1150).

z/OS *Návrh ukázky Get na systému z/OS*

Zde se dozvíte o návrhu ukázky Získat a o některých poznámkách k použití, které je třeba vzít v úvahu.

Průtok logikou programu je:

1. Připojte se ke správci front pomocí volání MQCONN. Pokud se toto volání nezdaří, vytiskněte kódy dokončení a příčiny a zastavte zpracování.

   Poznámka: Spouštíte-li ukázku v prostředí produktu CICS , nemusíte zadávat volání MQCONN; pokud ano, vrátí hodnotu DEF_HCONN. Pro následující volání MQI můžete použít manipulátor připojení MQHC_DEF_HCONN pro volání MQI.
2. Otevřete frontu pomocí volání MQOPEN s volbami MQOO_INPUT_SHARED a MQOO_BROWSE. Na vstupu do tohoto volání program používá popisovač připojení, který je vrácen v kroku “1” na stránce 1156. Pro strukturu deskriptoru objektu (MQOD) používá výchozí hodnoty pro všechna pole kromě pole názvu fronty, která se předává programu jako parametr programu. Pokud se volání MQOPEN nezdaří, vytiskněte kódy dokončení a příčiny a zastavte zpracování.
3. Vytvořte smyčku v rámci programu, který vydává volání MQGET, dokud se nenačte požadovaný počet zpráv z fronty. Pokud se volání MQGET nezdaří, je smyčka opuštěna předčasně, nedojde k pokusu o další volání MQGET a jsou vráceny kódy dokončení a příčiny. Ve volání MQGET jsou zadány následující volby:

   - MQGMO_NO_WAIT
   - ZPRÁVA MQGMO_ACCEPT_TRUNCATED_MESSAGE
   - MQGMO_SYNCPOINT nebo MQGMO_NO_SYNCPOINT
   - MQGMO_BROT FIRST a MQGMO_BRONEXT

   Popis těchto voleb naleznete v části MQGET. Pro každou zprávu se vytiskne číslo zprávy následované délkou zprávy a daty zprávy.
4. Zavřete frontu s použitím volání MQCLOSE s manipulátorem objektu vráceným v kroku “2” na stránce 1156. Pokud toto volání selže, vytiskněte kódy dokončení a příčiny.
5. Odpojte se od správce front s použitím volání MQDISC s manipulátorem připojení vráceným v kroku “1” na stránce 1156. Pokud toto volání selže, vytiskněte kódy dokončení a příčiny.

Poznámka: Pokud spouštíte ukázku v prostředí produktu CICS , není třeba volat volání MQDISC.

Poznámky k použití

- Chcete-li uchovat ukázky jednoduché, existují menší funkční rozdíly mezi jazykovými verzemi. Tyto rozdíly jsou však minimalizovány, pokud použijete rozvržení parametrů zobrazených v ukázce spuštění JCL, CSQ4BCJRa CSQ4BVJR,. Žádný z rozdílů se netýká rozhraní MQI.
- CSQ4BCJ1 vám umožňuje zadat více než čtyři číslice pro počet načtených zpráv.
- Zprávy delší než 64 kB jsou zkráceny.
- CSQ4BCJ1 může pouze správně zobrazovat znakové zprávy, protože se zobrazí pouze, dokud se nezobrazí první znak NULL (\0).
- Pro číselné pole počtu zpráv zadejte libovolnou číslici v rozsahu od 1 do 9999. Hodnota, kterou zadáte, by měla být kladné číslo. Chcete-li například získat jednu zprávu, můžete jako hodnotu zadat 1, 01, 001 nebo 0001. Zadáte-li nečíselné nebo záporné hodnoty, může dojít k chybě. Zadáte-li například hodnotu -1, program v jazyce COBOL načte jednu zprávu, ale program v C nenačte žádné zprávy.
- Pro oba programy, CSQ4BCJ1 a CSQ4BVJ1, zadejte B do parametru get, + + GET + +, chcete-li procházet zprávy.
- Pro oba programy, CSQ4BCJ1 a CSQ4BVJ1, zadejte S v parametru syncpoint, + + SYNC + +, pro zprávy, které mají být načteny v synchronizačním bodu.

z/OS *Ukázky získání pro prostředí dávky na systému z/OS*

Chcete-li ukázky spustit, upravte a spusťte ukázku JCL, jak je popsáno v tématu “Příprava a spuštění ukázkových aplikací pro dávkové prostředí v systému z/OS” na stránce 1145.

Programy používají následující parametry v EXEC PARM, oddělené mezerami v C a čárek v COBOLu:

1. Název správce front (4 znaky)
2. Název cílové fronty (48 znaků)
3. Počet zpráv k získání (až 4 číslice)
4. Volba pro prohlížení/získání zprávy (1 znak: B pro procházení nebo D pro destruktivním získání zpráv)
5. Ovládací prvek synchronizačního bodu (1 znak: S pro synchronizační bod nebo N pro žádný synchronizační bod)

Zadáte-li některý z těchto parametrů nesprávně, obdržíte příslušné chybové zprávy.

Výstup z ukázek je zapsán do datové sady SYSPRINT:

```
====================================
PARAMETERS PASSED :
QMGR    - VC9
QNAME    - A.Q
NUMMSGS   - 000000002
GET     - D
SYNCPOINT  - N
====================================
MQCONN SUCCESSFUL
MQOPEN SUCCESSFUL
000000000 : 000000010 : **********
000000001 : 000000010 : **********
000000002 MESSAGES GOT FROM QUEUE
MQCLOSE SUCCESSFUL
MQDISC SUCCESSFUL
```

z/OS *Ukázky získání pro prostředí produktu CICS na systému z/OS*

Speciální aspekty pro získání ukázek pro prostředí produktu CICS .

Transakce mají následující parametry v EXEC PARM, oddělené čárkami:

1. Počet zpráv, které se mají dostat (až čtyři číslice)

2. Volba pro prohlížení/získání zprávy (jeden znak: B pro procházení nebo D pro destruktivním získání zpráv)
3. Ovládací prvek synchronizačního bodu (jeden znak: S pro synchronizační bod nebo N pro žádný synchronizační bod)
4. Název cílové fronty (48 znaků)

Zadáte-li některý z těchto parametrů nesprávně, obdržíte příslušné chybové zprávy.

Pro ukázku jazyka COBOL vyvolejte ukázku Get v prostředí produktu CICS zadáním následujícího příkazu:

```
MVGT,9999,B,S,QUEUE.NAME
```

Pro ukázku C vyvolejte ukázku Get v prostředí produktu CICS zadáním:

```
MCGT,9999,B,S,QUEUE.NAME
```

Jsou-li zprávy načítány z fronty, jsou vloženy do fronty dočasného úložiště CICS se stejným názvem jako transakce CICS (například MCGT pro ukázku jazyka C).

Zde je příklad výstupu ukázek Get:

```
**************************** TOP OF QUEUE ************************
000000000 : 000000010: **********
000000001 : 000000010 :**********
**************************** BOTTOM OF QUEUE **********************
```

Poznámky k použití

- Chcete-li uchovat ukázky jednoduché, existují menší funkční rozdíly mezi jazykovými verzemi. Žádný z rozdílů se netýká rozhraní MQI.
- Pokud zadáte název fronty delší než 48 znaků, jeho délka je oříznuta na maximálně 48 znaků, ale nevrátí se žádná chybová zpráva.
- Před zadáním transakce stiskněte klávesu CLEAR.
- CSQ4CCJ1 může pouze správně zobrazovat znakové zprávy, protože se pouze zobrazí, dokud se nezobrazí první znak NULL (\0).
- Pro číselné pole zadejte jakékoli číslo v rozsahu od 1 do 9999. Hodnota, kterou zadáte, by měla být kladné číslo. Chcete-li například získat jednu zprávu, můžete zadat hodnotu 1, 01, 001 nebo 0001. Zadáte-li nečíselnou nebo zápornou hodnotu, může dojít k chybě.
- Zprávy delší než 24 526 bajtů v C a 9 950 bajtů v COBOLu jsou oříznuty. Důvodem je to, jak se používají dočasné paměťové fronty CICS .
- U obou programů, CSQ4CCK1 a CSQ4CVK1, zadejte B do parametru get, chcete-li procházet zprávy, jinak zadejte D. To provádí destruktivní volání MQGET. Zadáte-li jakoukoli jinou hodnotu, obdržíte chybovou zprávu.
- Pro oba programy, CSQ4CCJ1 a CSQ4CVJ1, zadejte S v parametru syncpoint k načtení zpráv v synchronizačním bodu. Zadáte-li hodnotu N v parametru synchronizačního bodu, budou volání MQGET vydána mimo synchronizační bod. Zadáte-li jakoukoli jinou hodnotu, obdržíte chybovou zprávu.

z/OS *Ukázka procházení v systému z/OS*

Ukázka Procházet je dávková aplikace, která demonstruje, jak procházet zprávy ve frontě pomocí volání MQGET.

Kroky aplikace se provedou všemi zprávami ve frontě a vytisknou prvních 80 bajtů každé z nich. Tuto aplikaci můžete použít k zobrazení zpráv ve frontě, aniž byste je měnili.

Zdrojové programy a ukázka kódu JCL spuštění jsou dodávány v jazycích COBOL, assembler, PL/I a C (viz Tabulka 162 na stránce 1146).

Chcete-li spustit aplikaci, upravte a spusťte ukázku spuštění JCL, jak je popsáno v tématu “Příprava a spuštění ukázkových aplikací pro dávkové prostředí v systému z/OS” na stránce 1145. Můžete se podívat na zprávy na jedné z vašich vlastních front zadáním názvu fronty v souboru JCL spuštění.

Když spustíte aplikaci (a ve frontě jsou nějaké zprávy), bude výstupní datová sada vypadat takto:

```
07/12/1998            SAMPLE QUEUE REPORT     PAGE  1
QUEUE MANAGER NAME : VC4
QUEUE NAME : CSQ4SAMP.DEAD.QUEUE
RELATIVE
MESSAGE  MESSAGE
NUMBER  LENGTH ------------------- MESSAGE DATA -------------

1    740 HELLO. PLEASE CALL ME WHEN YOU GET BACK.
2    429 CSQ4BQRM
3    429 CSQ4BQRM
4    429 CSQ4BQRM
5    22 THIS IS A TEST MESSAGE
6     8 CSQ4TEST
7    36 CSQ4MSG - ANOTHER TEST MESSAGE.....
!8     9 CSQ4STOP
********** END OF REPORT **********
```

Pokud ve frontě nejsou žádné zprávy, datová sada obsahuje pouze záhlaví a zprávy `Konec zprávy` . Dojde-li k chybě s některou z volání MQI, budou do datové sady výstupu přidány kódy dokončení a příčiny.

z/OS *Návrh ukázky procházení v systému z/OS*

Ukázková aplikace Procházet používá jediný programový modul; jeden z podporovaných programovacích jazyků je k dispozici v každém z podporovaných programovacích jazyků.

Průtok logikou programu je:

1. Otevřete tiskovou datovou sadu a vytiskněte řádek s titulkem sestavy. Zkontrolujte, zda byly názvy správce front a fronty předány ze souboru JCL spuštění. Pokud byly předány obě jména, vytiskněte si řádky sestavy, které obsahují jména. Pokud tomu tak není, vytiskněte chybovou zprávu, zavřete tiskovou datovou sadu a zastavte zpracování.

   Způsob, jakým program testuje parametry, které je předáván z JCL, závisí na jazyku, ve kterém je program napsán. Další informace viz “Aspekty návrhu závislých na jazyku v produktu z/OS” na stránce 1160.
2. Připojte se ke správci front pomocí volání MQCONN. Pokud toto volání není úspěšné, vytiskněte kódy dokončení a příčiny, zavřete tiskovou datovou sadu a zastavte zpracování.
3. Otevřete frontu s použitím volání MQOPEN s volbou MQOO_BROWSE. Na vstupu do tohoto volání program používá popisovač připojení vrácený v kroku “2” na stránce 1159. Pro strukturu deskriptoru objektu (MQOD) používá výchozí hodnoty pro všechna pole kromě názvu fronty (která byla předána v kroku “1” na stránce 1159). Pokud toto volání není úspěšné, vytiskněte kódy dokončení a příčiny, zavřete tiskovou datovou sadu a zastavte zpracování.
4. Procházejte první zprávu ve frontě pomocí volání MQGET. Na vstupu do tohoto volání program uvádí:
   - Připojení a obslužné rutiny front z kroků “2” na stránce 1159 a “3” na stránce 1159
   - Struktura MQMD se všemi poli nastavními na počáteční hodnoty
   - Dvě volby:
     - NEJPRVE MQGMO_BROWSE_FIRST
     - SOUBOR MQGMO_ACCEPT_TRUNCATED_MSG
   - Vyrovnávací paměť o velikosti 80 bajtů, která uchovává data zkopírovaná ze zprávy

   Volba MQGMO_ACCEPT_TRUNCATED_MSG umožňuje dokončení volání i v případě, že zpráva je delší než 80bajtová vyrovnávací paměť zadaná ve volání. Pokud je zpráva delší než vyrovnávací paměť, zpráva se osekne tak, aby se vešla do vyrovnávací paměti, a jsou nastaveny kódy dokončení a příčiny, aby se tato zpráva zobrazila. Vzorek byl navržen tak, aby zprávy byly zkráceny na 80 znaků, aby se

sestava snadno čila. Velikost vyrovnávací paměti je nastavena příkazem `DEFINE` , takže ji můžete snadno změnit, pokud chcete.

5. Proveďte následující smyčku, dokud se volání MQGET nezdaří:
   a. Tisknout řádek sestavy zobrazující:
      - Pořadové číslo zprávy (jedná se o počet operací procházení).
      - Skutečná délka zprávy (nikoli zkrácená délka). Tato hodnota je vrácena v poli `DataLength` volání MQGET.
      - Prvních 80 bajtů dat zprávy.
   b. Vynulovat pole `MsqId` a `CorrelId` struktury MQMD na hodnoty null.
   c. Projděte další zprávu pomocí volání MQGET s těmito dvěma volbami:
      - PŘÍŠTĚ MQGMO_BROWSE_NEXT
      - SOUBOR MQGMO_ACCEPT_TRUNCATED_MSG
6. Pokud se volání MQGET nezdaří, otestujte kód příčiny a zjistěte, zda došlo k selhání volání, protože kurzor procházení se dostal na konec fronty. V takovém případě vytiskněte zprávu `Konec zprávy` a přejděte na krok “7” na stránce 1160 . v opačném případě vytiskněte kódy dokončení a příčiny, zavřete tiskovou datovou sadu a zastavte zpracování.
7. Zavřete frontu s použitím volání MQCLOSE s manipulátorem objektu vráceným v kroku “3” na stránce 1159.
8. Odpojte se od správce front s použitím volání MQDISC s manipulátorem připojení vráceným v kroku “2” na stránce 1159.
9. Zavřete sadu tiskových dat a zastavte zpracování.

z/OS *Aspekty návrhu závislých na jazyku v produktu z/OS*

Zdrojové moduly jsou k dispozici pro ukázku Procházet ve čtyřech programovacích jazycích.

Mezi zdrojové moduly existují dva hlavní rozdíly:

- Při testování parametrů předaných ze spuštění JCL se moduly jazyka COBOL, PL/I a assembler-Language vyhledávají pro čárku (,). Pokud kód JCL projde `PARM=(,LOCALQ1)` , pokusí se aplikace otevřít frontu LOCALQ1 ve výchozím správci front. Pokud za čárku (nebo bez čárky) neexistuje žádný název, aplikace vrátí chybu. Modul C nevyhledává znak čárky. Pokud kód JCL předá jeden parametr (například `PARM='LOCALQ1'`), použije modul jazyka C toto jako název fronty ve výchozím správci front.
- Chcete-li modul v asembleru použít jednoduchý modul, použije formát data *rr/ddd* (například 05/116) při vytváření tiskové sestavy. Ostatní moduly používají kalendářní datum ve formátu *mm/dd/rr* .

z/OS *Ukázka tiskové zprávy v systému z/OS*

Ukázka Print Message je dávková aplikace, která demonstruje, jak odebrat všechny zprávy z fronty pomocí volání MQGET.

Ukázka Tisková zpráva používá tři parametry:

1. Název správce front
2. Název zdrojové fronty
3. Nepovinný parametr pro vlastnosti

Tiskne také pro každou zprávu pole deskriptoru zpráv následovaná daty zprávy. Program tiskne data v hexadecimálním formátu i ve formě znaků (pokud jsou tisknutelné). Pokud znak není tisknutelný, nahradí jej znakem tečky (.). Tento program můžete použít při diagnostice problémů s aplikací, která vkládá zprávy do fronty.

Přípustné hodnoty pro parametr vlastnosti jsou:

Tabulka 174. Přípustné hodnoty pro parametr vlastnosti	
Hodnota	**Chování**
0	Výchozí chování, jak bylo pro IBM WebSphere MQ 6. Vlastnosti, které se doručí do aplikace, závisí na atributu fronty produktu **PropertyControl** , ze kterého se zpráva načítá.

1	Popisovač zprávy je vytvořen a použit s parametrem MQGET. Vlastnosti zprávy s výjimkou těch, které jsou obsaženy v deskriptoru (či rozšíření) zprávy, jsou zobrazeny podobným způsobem jako deskriptor zprávy. Příklad:
	`****Message properties****`
	`  property name: property value`
	Nebo nejsou-li k dispozici žádné vlastnosti:
	`****Message properties****`
	`  None`
	Číselné hodnoty jsou zobrazeny pomocí `printf`, hodnoty řetězce jsou uzavřeny v jednoduchých uvozovkách a bajtové řetězce jsou obklopeny X a jednoduchými uvozovkami, jako pro deskriptor zprávy.

2	Hodnota MQGMO_NO_PROPERTIES je určena tak, aby byly vráceny pouze vlastnosti deskriptoru zpráv.
3	Je zadán parametr MQGMO_PROPERTIES_FORCE_MQRFH2 , takže všechny vlastnosti jsou vráceny v datech zprávy.
4	Funkce MQGMO_PROPERTIES_COMPATIBILITY je určena tak, aby všechny vlastnosti mohly být vráceny v závislosti na tom, zda je vlastnost IBM WebSphere MQ 6 zahrnuta, jinak jsou vlastnosti zrušeny.

Můžete změnit aplikaci tak, aby procházeli zprávy, spíše než aby je odebírane z fronty. Chcete-li tuto akci provést, zkompilujte s volbou -DBROWSE, abyste definovali makro BROWSE, jak je uvedeno v “Návrh ukázky tiskové zprávy v systému z/OS” na stránce 1162. Spustitelný kód je poskytnut pro vás v knihovně SCSQLOAD. Modul CSQ4BCG0 je sestaven s modulem -DBROWSE; modul CSQ4BCG1 destruktivně čte frontu.

Aplikace má jediný zdrojový program, který je napsán v jazyce C. Ukázka kódu JCL spuštění je také dodána (viz Tabulka 163 na stránce 1146).

Chcete-li spustit aplikaci, upravte a spusťte ukázku spuštění JCL, jak je popsáno v tématu “Příprava a spuštění ukázkových aplikací pro dávkové prostředí v systému z/OS” na stránce 1145. Když spustíte aplikaci (a ve frontě jsou nějaké zprávy), bude výstupní datová sada vypadat jako v produktu Obrázek 151 na stránce 1162.

```
CSQ4BCG1 - starts here
**********************

MQCONN to MQ1E
MQOPEN - 'TEST.QUEUE'
MQCRTMH


MQGET of message number 1
****Message descriptor****
StrucId : 'MD ' Version : 2
Report  : 0 MsgType : 8
Expiry  : -1 Feedback : 0
Encoding : 785 CodedCharSetId : 500
Format : 'MQSTR  '
Priority : 0 Persistence : 0
MsgId : X'C3E2D840D4D8F1C540404040404040C1EA537F03167D88'
CorrelId : X'C3E2D840D4D8F1C540404040404040C1EA537F0317A928'
BackoutCount : 0
ReplyToQ    : '                          '
ReplyToQMgr  : ''
** Identity Context
UserIdentifier : 'FRED    '
AccountingToken :
X'0000000000000000000000000000000000000000000000000000000000000000'
ApplIdentityData : '                 '
** Origin Context
PutApplType  : '2'
PutApplName  : 'FRED6            '
PutDate : '20080207'  PutTime : '17373745'
ApplOriginData : '  '
GroupId : X'000000000000000000000000000000000000000000000000'
MsgSeqNumber  : '1'
Offset     : '0'
MsgFlags    : '0'
OriginalLength : '-1'

****Message properties****
None

****  Message   ****

length - 30 bytes

00000000: E388 89A2 4089 A240 8140 A289 9497 9385 'This is a simple'
00000010: 40A3 85A2 A340 9485 A2A2 8187 855A   ' test message! '



No more messages
MQDLTMH
MQCLOSE
MQDISC
```

Obrázek 151. Příklad sestavy z ukázkové aplikace Tisková zpráva

z/OS *Návrh ukázky tiskové zprávy v systému z/OS*

Ukázková aplikace Print Message používá jediný program zapsaný v jazyce C.

Průtok logikou programu je:

1. Zkontrolujte, zda byly názvy správce front a fronty předány ze souboru JCL spuštění. Pokud tomu tak není, vytiskněte chybovou zprávu a zastavte zpracování.
2. Připojte se ke správci front pomocí volání MQCONN. Není-li toto volání úspěšné, vytiskněte kódy dokončení a příčiny a zastavte zpracování; v opačném případě vytiskněte název správce front.
3. Otevřete frontu s použitím volání MQOPEN s volbou MQOO_INPUT_SHARED.

   Poznámka: Chcete-li, aby aplikace procházela zprávy místo jejich odebrání z fronty, zkompilujte ukázku pomocí parametru -DBROWSE nebo přidejte na začátek zdroje položku `#define BROWSE` . Když to provedete, preprocesor makra přidá řádek do programu, který vybere volbu MQOO_BROWSE v kompilaci.

Na vstupu do tohoto volání program používá popisovač připojení vrácený v kroku “2” na stránce 1162. Pro strukturu deskriptoru objektu (MQOD) používá výchozí hodnoty pro všechna pole kromě názvu fronty (která byla předána v kroku “1” na stránce 1162). Není-li toto volání úspěšné, vytiskněte kódy dokončení a kódy příčiny a ukončete zpracování; v opačném případě vytiskněte název fronty.

4. Pokud použijete popisovač zpráv k získání vlastností zprávy, použijte MQCRTMH k vytvoření takového popisovače pro použití s následnými voláními MQGET. Pokud toto volání není úspěšné, vytiskněte kódy dokončení a příčiny a zastavte zpracování.
5. Nastavte volby příkazu get tak, aby odrážely akce požadavku pro všechny vlastnosti zprávy.
6. Proveďte následující smyčku, dokud se volání MQGET nezdaří:
   a. Inicializujte vyrovnávací paměť na prázdné místo, aby data zprávy nebyla porušena žádnými daty, která jsou již ve vyrovnávací paměti.
   b. Nastavte pole `MsgId` a `CorrelId` struktury MQMD na hodnoty null tak, aby volání MQGET vybíraly první zprávu z fronty.
   c. Získejte zprávu z fronty pomocí volání MQGET. Na vstupu do tohoto volání program uvádí:
      - Připojení a obslužné rutiny objektu z kroků “2” na stránce 1162 a “3” na stránce 1162.
      - Struktura MQMD se všemi poli nastavními na počáteční hodnoty. (`MsgId` a `CorrelId` jsou vynulovány na hodnoty null pro každé volání MQGET.)
      - Volba MQGMO_NO_WAIT.

        Poznámka: Chcete-li, aby aplikace procházela zprávy místo jejich odebrání z fronty, zkompilujte ukázku pomocí parametru -DBROWSE nebo přidejte na začátek zdroje položku `#define BROWSE` . Pokud to provedete, preprocesor makra přidá řádek do programu, který ke kompilaci vybere volbu MQGMO_BROWSE NEXT. Je-li tato volba použita ve volání proti frontě, pro kterou nebyl dříve použit kurzor procházení s aktuálním popisovačem objektu, je kurzor procházení umístěn logicky před první zprávou.
      - Vyrovnávací paměť o velikosti 64KB , která uchovává data zkopírovaná ze zprávy.
   d. Zavolejte subrutinu printMD . Tím se vytiskne název každého pole v deskriptoru zpráv následován jeho obsahem.
   e. Pokud jste vytvořili obslužnou rutinu zpráv v kroku “4” na stránce 1163 , vyvolejte subrutinu printProperties , aby se zobrazily všechny vlastnosti zprávy.
   f. Tiskne délku zprávy a za ní následují data zprávy. Každý řádek dat zprávy je v tomto formátu:
      - Relativní pozice (hexadecimálně) této části dat
      - 16 bajtů hexadecimálních dat
      - Tentýž 16 bajtů dat ve znakovém formátu, je-li tisknutelný (netisknutelné znaky se nahradí tečkami)
7. Pokud se volání MQGET nezdaří, otestujte kód příčiny a zjistěte, zda se volání nezdařilo, protože ve frontě již nejsou žádné další zprávy. V tomto případě vytisknete zprávu: Žádné další zprávy; jinak vytiskněte kódy dokončení a příčiny. V obou případech přejděte na krok “9” na stránce 1163.

   Poznámka: Volání MQGET selže, pokud najde zprávu, která má více než 64KB dat. Chcete-li změnit program pro zpracování větších zpráv, můžete provést jednu z následujících možností:
   - Přidejte volbu MQGMO_ACCEPT_TRUNCATED_MSG do volání MQGET, takže volání získá první 64KB dat a zahodí zbytek
   - Učinit program zanechat zprávu ve frontě, když nalezne jeden s tímto množstvím dat
   - Zvětšit velikost vyrovnávací paměti
8. Pokud jste vytvořili obslužnou rutinu zpráv v kroku “4” na stránce 1163 , zavolejte MQDLTMH, abyste ji odstranili.
9. Zavřete frontu s použitím volání MQCLOSE s manipulátorem objektu vráceným v kroku “3” na stránce 1162.

10. Odpojte se od správce front s použitím volání MQDISC s manipulátorem připojení vráceným v kroku “2” na stránce 1162.

z/OS *Ukázka atributů fronty v systému z/OS*

Ukázka Atributy fronty je dialogová aplikace CICS , která demonstruje použití volání MQINQ a MQSET.

Ukazuje, jak se dotázat na hodnoty atributů **InhibitPut** a **InhibitGet** front a jak je změnit, aby programy nemohly vkládat zprávy do fronty nebo získávat zprávy z fronty. Při testování programu můžete chtít *zamknout* frontu tímto způsobem.

Aby se zabránilo náhodnému interferenci s vlastními frontami, tato ukázka pracuje pouze na objektu fronty, který má znaky CSQ4SAMP v prvních osmi bajtech jeho názvu. Zdrojový kód však obsahuje komentáře, které vám ukážou, jak odstranit toto omezení.

Zdrojové programy jsou dodávány v jazycích COBOL, assembler a C (viz Tabulka 169 na stránce 1151).

Verze v asembleru-language ukázky používá znovu zadaný kód. Chcete-li tak učinit, všimnete si, že kód pro každé volání MQI v této verzi ukázky obsahuje klíčové slovo MF, například:

```
CALL MQCONN,(NAME,HCONN,COMPCODE,REASON),MF=(E,PARMAREA),VL
```

(Klíčové slovo VL znamená, že pro ladění programu můžete použít poskytovanou transakci CEDF (Execution Diagnostic Facility) produktu CICS .) Další informace o psaní opakovaných programů najdete v tématu Kódování v jazyce assembler v systému System/390.

Chcete-li spustit aplikaci, spusťte svůj systém CICS a použijte následující transakce CICS :

- Pro COBOL, MVC1
- Pro jazyk assembler, MAC1
- Pro C, MCC1

Název kterékoli z těchto transakcí můžete změnit změnou datové sady CSD uvedené v kroku 3.

Návrh vzorku

Když ukázku spustíte, zobrazí se mapa obrazovky, která obsahuje pole pro:

- Název fronty
- Požadavek uživatele (platné akce jsou: dotázat, povolit nebo zakázat)
- Aktuální stav operací vložení pro frontu
- Aktuální stav operací získání pro frontu

První dvě pole jsou určena pro vstup uživatele. Poslední dvě pole jsou vyplněna aplikací: zobrazují slovo `INHIBITED` nebo slovo `ALLOWED`.

Aplikace ověří hodnoty, které jste zadali v prvních dvou polích. Kontroluje, zda název fronty začíná znaky CSQ4SAMP a že jste zadali jeden ze tří platných požadavků v poli Akce. Aplikace převede všechny vaše vstupní hodnoty na velká písmena, takže nemůžete používat žádné fronty s názvy, které obsahují malá písmena.

Zadáte-li do pole **Akce** hodnotu `inquire` , bude průtok programovou logikou:

1. Otevřete frontu pomocí volání MQOPEN s volbou MQOO_INQUIRE.
2. Volejte MQINQ s použitím selektorů MQIA_INHIBIT_GET a MQIA_INHIBIT_PUT
3. Zavřít frontu pomocí volání MQCLOSE
4. Analyzujte atributy, které jsou vráceny v parametru **IntAttrs** volání MQINQ, a podle potřeby přesuňte slova INHIBITED nebo ALLOWED na odpovídající pole obrazovky.

Zadáte-li do pole **Akce** hodnotu `inhibit` , bude průtok programovou logikou:

1. Otevřete frontu s použitím volání MQOPEN s volbou MQOO_SET

2. Volání MQSET pomocí selektorů MQIA_INHIBIT_GET a MQIA_INHIBIT_PUT a s hodnotami MQQA_GET_INHIBITED a MQQA_PUT_INHIBITED v parametru **IntAttrs**
3. Zavřít frontu pomocí volání MQCLOSE
4. Přesunout slovo INHIBITED do příslušných polí obrazovky

Zadáte-li do pole **Akce** hodnotu `allow` , aplikace provede podobné zpracování jako u požadavku na blokování. Jediné rozdíly jsou nastavení atributů a slova zobrazená na obrazovce.

Když aplikace otevře frontu, použije výchozí obslužnou rutinu připojení ke správci front. (CICS naváže spojení se správcem front při spuštění vašeho systému CICS .) Aplikace může v této fázi zachytit následující chyby:

- Aplikace není připojena ke správci front
- Fronta neexistuje.
- Uživatel nemá oprávnění pro přístup ke frontě
- Aplikace nemá autorizaci k otevření fronty.

Pro jiné chyby MQI aplikace zobrazuje kódy dokončení a příčiny.

z/OS *Ukázka programu Mail Manager v systému z/OS*

Ukázková aplikace Správce pošty je sada programů, které demonstrují odesílání a příjem zpráv v rámci jednoho prostředí i v různých prostředích. Aplikace je jednoduchý elektronický poštovní systém, který umožňuje uživatelům vyměňovat si zprávy, a to i v případě, že používají různé správce front.

Aplikace demonstruje, jak vytvářet fronty pomocí volání MQOPEN a vloží příkazy IBM MQ for z/OS do vstupní fronty systémového příkazu.

K dispozici jsou tři verze aplikace:

- Aplikace CICS napsaná v jazyce COBOL
- Aplikace TSO napsaná v jazyce COBOL
- Aplikace TSO napsaná v jazyce C

z/OS *Příprava ukázky programu Mail Manager v systému z/OS*

Program Mail Manager je poskytován ve verzích, které jsou spuštěny ve dvou prostředích. Příprava, kterou musíte provést před spuštěním aplikace, závisí na prostředí, které chcete použít.

Uživatelé mohou přistupovat k frontám pošty a frontám přezdívek z TSO a CICS tak dlouho, jak se jejich přihlašovací ID uživatelů shodují na každém systému.

Než budete moci odesílat zprávy do jiného správce front, je třeba do tohoto správce front nastavit kanál zpráv. Chcete-li to provést, použijte funkci řízení kanálu produktu IBM MQpopsanou v části Funkce řízení kanálů.

Příprava ukázky pro prostředí TSO

Postupujte takto:

1. Připravte ukázku tak, jak je popsáno v části “Příprava ukázkových aplikací pro prostředí TSO v systému z/OS” na stránce 1147.
2. Tailor a CLIST, které jsou k dispozici pro vzorek k definování:
   - Umístění panelů
   - Umístění souboru zpráv
   - Umístění zaváděcích modulů
   - Název správce front, který chcete použít spolu s aplikací.

   Samostatná CLIST je poskytována pro každou jazykovou verzi vzorku:
   - Pro verzi jazyka COBOL: CSQ4RVD1

- Pro verzi C: CSQ4RCD1

3. Ujistěte se, že fronty použité aplikací jsou k dispozici ve správci front. (Fronty jsou definovány v CSQ4CVD.)

Poznámka: VS COBOL II nepodporuje souběžné zpracování více úloh s ISPF. To znamená, že nemůžete použít ukázkovou aplikaci Správce pošty na obou stranách rozdělené obrazovky. Pokud ano, výsledky jsou nepředvídatelné.

z/OS *Spuštění ukázky programu Mail Manager v systému z/OS*

Chcete-li spustit ukázku v prostředí CICS Transaction Server pro prostředí z/OS , spusťte transakci MAIL. Pokud jste se dosud nepřihlásili k produktu CICS, aplikace vás vyzve k zadání ID uživatele, do kterého může odeslat vaši poštu.

Spustíte-li aplikaci, otevře se poštovní fronta. Pokud tato fronta neexistuje, aplikace ji pro vás vytvoří. Poštovní fronty mají názvy ve formátu CSQ4SAMP.MAILMGR. *userid*, kde *userid* závisí na prostředí:

V TSO
ID TSO uživatele

VeCICS
Přihlášení uživatele serveru CICS nebo ID uživatele zadané uživatelem při výzvě, kdy byl spuštěn správce pošty.

Všechny části názvů front, které používá správce pošty, musí být velkými písmeny.

Aplikace pak zobrazí panel nabídky, který má volby pro:

- Číst příchozí poštu
- Odeslat e-mail
- Vytvořit přezdívku

Panel nabídky také zobrazuje, kolik zpráv čeká ve vaší poštovní frontě. Každá z voleb nabídky zobrazí další panel:

Číst příchozí poštu
Správce pošty zobrazí seznam zpráv, které se nacházejí ve vaší poštovní frontě. (Zobrazí se pouze prvních 99 zpráv ve frontě.) Příklad tohoto panelu viz Obrázek 154 na stránce 1170. Vyberete-li zprávu z tohoto seznamu, zobrazí se obsah zprávy (viz Obrázek 155 na stránce 1171).

Odeslat e-mail
Panel vás vyzve k zadání:

- Jméno uživatele, kterému chcete odeslat zprávu
- Název správce front, který vlastní jejich poštovní frontu.
- Text zprávy

Do pole jména uživatele můžete zadat buď ID uživatele, nebo přezdívku, kterou jste vytvořili pomocí programu Mail Manager. Pole názvu správce front můžete ponechat prázdné, je-li fronta pošty uživatele vlastněna stejným správcem front, kterého používáte, a pokud jste do pole jména uživatele zadali přezdívku, musíte ji ponechat prázdné:

- Uvedete-li pouze jméno uživatele, program nejprve předpokládá, že jméno je přezdívka, a odešle zprávu objektu definovanému tímto jménem. Pokud taková přezdívka neexistuje, program se pokusí odeslat zprávu do lokální fronty s tímto názvem.
- Zadáte-li jméno uživatele i jméno správce front, program odešle zprávu do poštovní fronty, která je definována těmito dvěma názvy.

Chcete-li například odeslat zprávu uživateli JONESM ve vzdáleném správci front QM12, můžete jim odeslat zprávu jedním z následujících dvou způsobů:

- Použijte obě pole k uvedení uživatele JONESM ve správci front QM12.

- Nadefinujte přezdívku (např. MARY) pro tohoto uživatele a odešlete jim zprávu tím, že MARY vložíte do pole jména uživatele a nic do pole s názvem správce front.

Vytvořit přezdívku

Můžete definovat název pro east-zapamatování identity, který můžete použít, když posíláte zprávu jinému uživateli, kterého se často zkontaktujete. Budete vyzváni k zadání ID uživatele jiného uživatele a jména správce front, který vlastní jejich poštovní frontu.

Přezdívky jsou fronty, které mají názvy ve formě CSQ4SAMP.MAILMGR. *userid.nickname*, kde *userid* je vaše vlastní ID uživatele a *přezdívka* je přezdívka, kterou chcete použít. S názvy strukturovanými tímto způsobem mohou uživatelé mít každá svou vlastní sadu přezdívek.

Typ fronty, kterou program vytvoří, závisí na tom, jak dokončíte pole na panelu Vytvořit přezdívku:

- Pokud zadáte pouze jméno uživatele nebo název správce front je shodný s názvem správce front, ke kterému je správce pošty připojen, program vytvoří alias frontu.
- Pokud zadáte jak jméno uživatele, tak jméno správce front (a správce front není ten, ke kterému je správce pošty připojen), program vytvoří lokální definici vzdálené fronty. Program nekontroluje existenci fronty, na kterou je tento definice interpretována, nebo dokonce i existenci vzdáleného správce front.

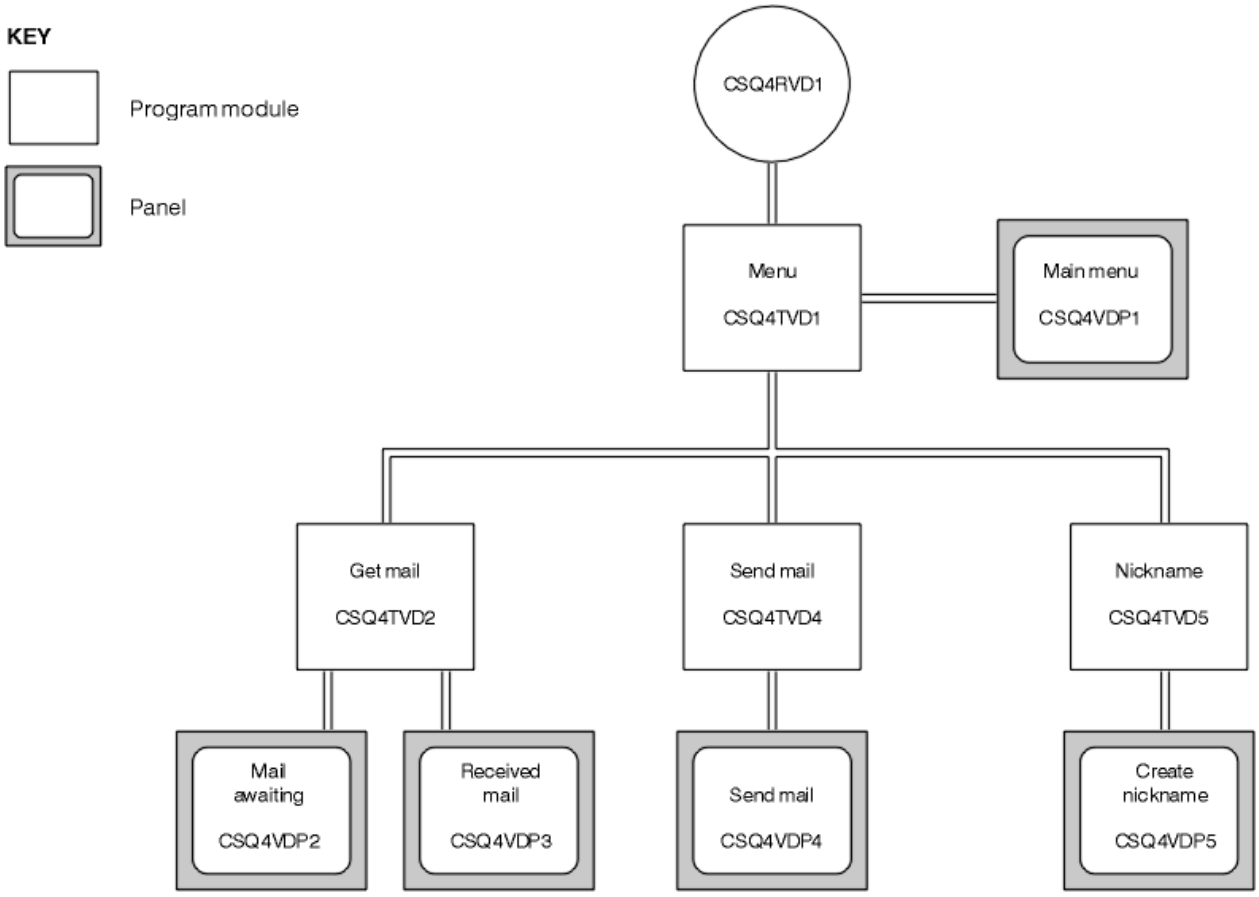
Je-li například vaším vlastním ID uživatele SMITHK a vytváříte přezdívku s názvem MARY pro uživatele JONESM (který používá vzdáleného správce front QM12), vytvoří inicializační program lokální definici vzdálené fronty s názvem CSQ4SAMP.MAILMGR.SMITHK.MARY. Tato definice se vyřeší na poštovní frontu Mary, která je CSQ4SAMP.MAILMGR.JONESM ve správci front QM12. Pokud používáte správce front QM12 sami, místo toho vytvoří alias frontu se stejným názvem (CSQ4SAMP.MAILMGR.SMITHK.MARY).

Verze C aplikace TSO umožňuje větší využití schopností zpracování zpráv ISPF, než je verze jazyka COBOL. Můžete si všimnout, že různé chybové zprávy jsou zobrazeny verzemi C a COBOL.

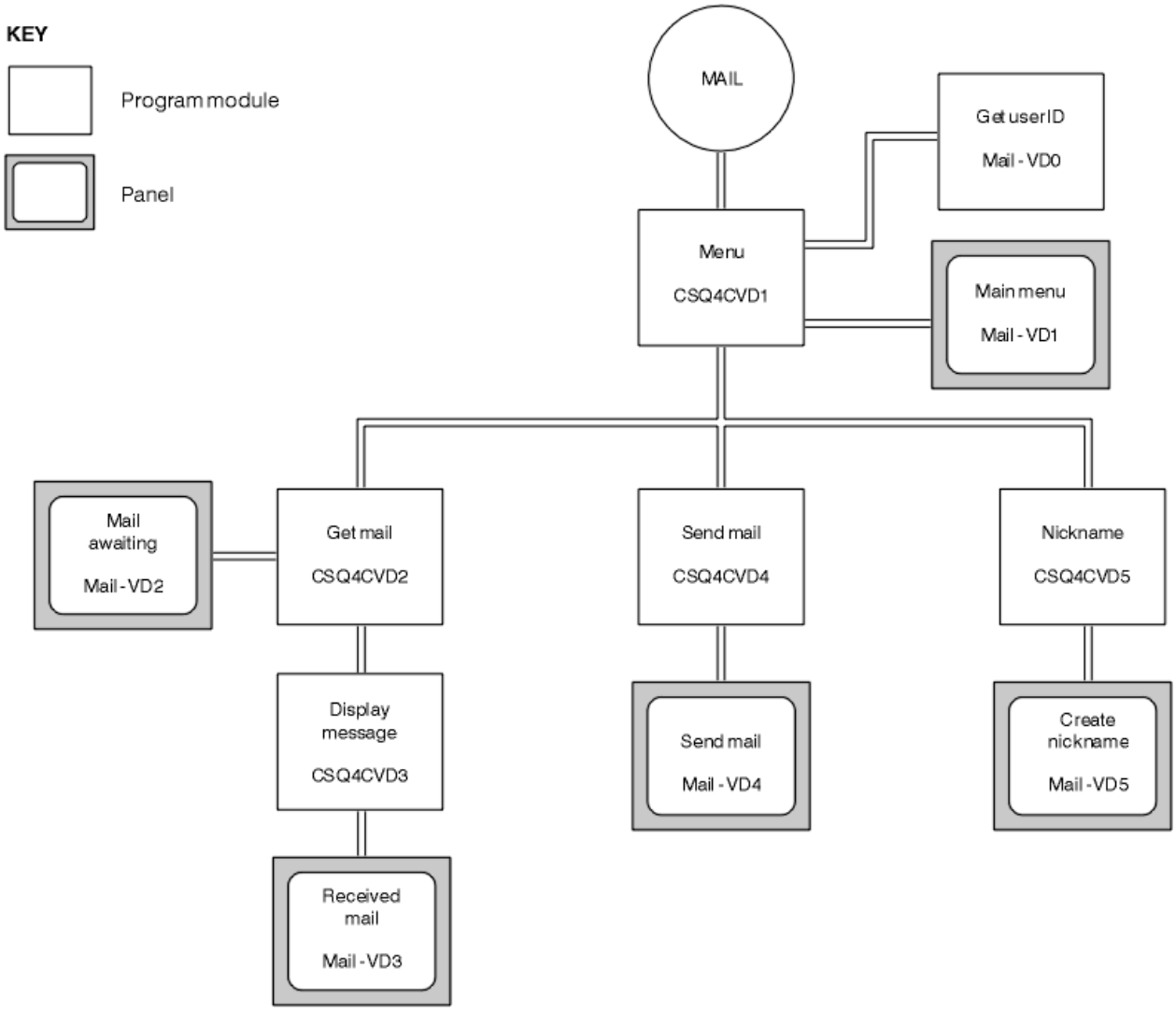
z/OS *Návrh ukázky programu Mail Manager v systému z/OS*

Následující sekce popisují každý z programů, které tvoří ukázkovou aplikaci Správce pošty.

Vztahy mezi programy a panely, které aplikace používá, jsou zobrazeny v produktu Obrázek 152 na stránce 1168 pro verzi TSO a Obrázek 153 na stránce 1169 pro produkt CICS Transaction Server pro verzi z/OS .



Obrázek 152. Programy a panely pro verze TSO správce Mail Manager



Obrázek 153. Programy a panely pro verzi produktu CICS správce pošty

z/OS *Program nabídky na systému z/OS*

V prostředí TSO se program nabídky vyvolá pomocí programu CLIST. V prostředí CICS je program vyvolán pomocí transakce MAIL.

Program v menu (CSQ4TVD1 for TSO, CSQ4CVD1 for CICS) je počáteční program v sadě. Zobrazí nabídku (CSQ4VDP1 pro TSO, VD1 pro CICS) a vyvolá další programy, když jsou vybrány z nabídky.

Program nejprve získá ID uživatele:

- Pokud se uživatel přihlásil do CICSverze programu CICS , je ID uživatele získáno pomocí příkazu CICS ASSIGN USERID. Pokud se uživatel nezapsal, program zobrazí přihlášení na panelu (CSQ4VD0), aby vyzval uživatele k zadání ID uživatele. V rámci tohoto programu není žádné zpracování zabezpečení; uživatel může zadat libovolné ID uživatele.
- V verzi TSO se ID uživatele získává z TSO v CLIST. Je předáván do programu nabídky jako proměnná ve sdíleném fondu ISPF.

Poté, co program získá ID uživatele, zkontroluje, zda má uživatel poštovní frontu (CSQ4SAMP.MAILMGR. *ID uživatele*). Pokud poštovní fronta neexistuje, program ji vytvoří vložením zprávy do vstupní fronty příkazu systému. Zpráva obsahuje příkaz IBM MQ for z/OS DEFINE QLOCAL. Definice objektu, kterou tento příkaz používá, nastavuje maximální hloubku fronty na 9999 zpráv.

Program také vytvoří dočasnou dynamickou frontu pro zpracování odpovědí ze vstupní fronty systémového příkazu. Za tímto účelem program používá volání MQOPEN a specifikuje

SYSTEM.DEFAULT.MODEL.QUEUE jako šablona pro dynamickou frontu. Správce front vytvoří dočasnou dynamickou frontu s názvem, který má předponu CSQ4SAMP; , zbývající část názvu vygeneruje správce front.

Program potom otevře uživatelovu poštovní frontu a zjistí počet zpráv ve frontě tím, že se dotazuje na aktuální hloubku fronty. Aby to bylo možné, program používá volání MQINQ a určuje selektor MQIA_CURRENT_Q_DEPTH.

Program pak provede smyčku, která zobrazí menu a zpracuje výběr, který uživatel provede. Smyčka se zastaví, když uživatel stiskne klávesu PF3 . Je-li proveden platný výběr, je spuštěn příslušný program; v opačném případě se zobrazí chybová zpráva.

z/OS *Získat-pošta a zobrazit-programy zpráv v systému z/OS*

Ve verzích TSO pro aplikaci jsou funkce get-mail a display-message prováděny pomocí stejného programu (CSQ4TVD2). Ve verzi CICS aplikace jsou tyto funkce prováděny oddělenými programy (CSQ4CVD2 a CSQ4CVD3).

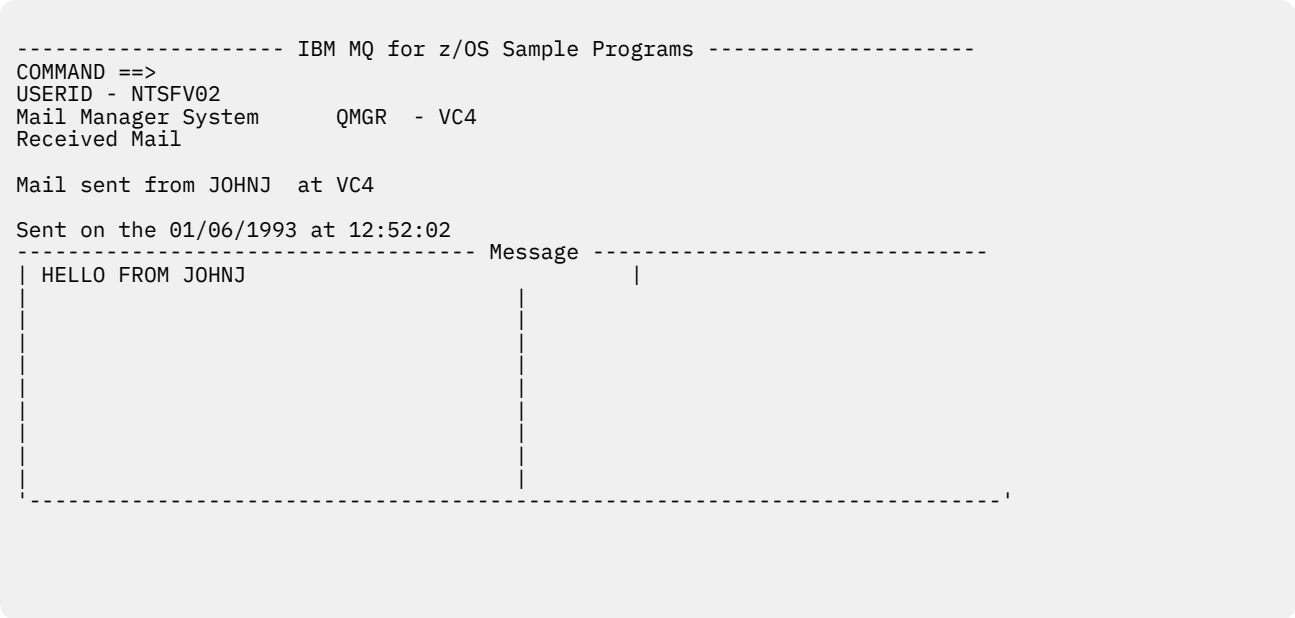
Čekající panel pošty (CSQ4VDP2 pro TSO, VD2 pro CICS ; viz Obrázek 154 na stránce 1170 .) zobrazuje všechny zprávy, které jsou ve frontě pošty uživatele. Chcete-li vytvořit tento seznam, program použije volání MQGET k procházení všech zpráv ve frontě a ukládá informace o každém z nich. Kromě zobrazených informací zaznamenává program zprávy `MsgId` a `CorrelId` jednotlivých zpráv.

```
---------------------- IBM MQ for z/OS Sample Programs ------- ROW 16 OF 29
COMMAND ==>                          Scroll  ===> PAGE
USERID - NTSFV02
Mail Manager System      QMGR  - VC4
Mail Awaiting

Msg   Mail    Date    Time
No    From    Sent    Sent
16
16   Deleted
17   JOHNJ   01/06/1993 12:52:02
18   JOHNJ   01/06/1993 12:52:02
19   JOHNJ   01/06/1993 12:52:03
20   JOHNJ   01/06/1993 12:52:03
21   JOHNJ   01/06/1993 12:52:03
22   JOHNJ   01/06/1993 12:52:04
23   JOHNJ   01/06/1993 12:52:04
24   JOHNJ   01/06/1993 12:52:04
25   JOHNJ   01/06/1993 12:52:05
26   JOHNJ   01/06/1993 12:52:05
27   JOHNJ   01/06/1993 12:52:05
28   JOHNJ   01/06/1993 12:52:06
29   JOHNJ   01/06/1993 12:52:06
```

Obrázek 154. Příklad panelu zobrazujícího seznam čekajících zpráv

Na panelu Čekání pošty může uživatel vybrat jednu zprávu a zobrazit její obsah (viz příklad Obrázek 155 na stránce 1171). Program používá volání MQGET k odstranění této zprávy z fronty pomocí `MsgId` a `CorrelId` , které program zaznamenal, když prohlédl všechny zprávy. Toto volání MQGET se provádí pomocí volby MQGMO_SYNCPOINT. Program zobrazí obsah zprávy a poté deklaruje synchronizační bod: Toto potvrdí volání MQGET, takže zpráva nyní již neexistuje.



Obrázek 155. Příklad panelu ukazující obsah zprávy

Funkce, kterou poskytuje Správce pošty, má možnost dát uživateli možnost ponechat zprávu ve frontě nechat po zobrazení jejího obsahu. Chcete-li to provést, musíte odvolat volání MQGET, které odebere zprávu z fronty po zobrazení zprávy.

z/OS *Odeslání-poštovní program v systému z/OS*

Když uživatel dokončil panel Odeslat poštu (CSQ4VDP4 for TSO, VD4 for CICS), program odesílání pošty (CSQ4TVD4 pro TSO, CSQ4CVD4 pro CICS) vloží zprávu do poštovní fronty příjemce.

K provedení tohoto úkolu program používá volání MQPUT1 . Místo určení zprávy závisí na tom, jak uživatel vyplnil pole na panelu Odeslat poštu:

- Pokud uživatel zadal pouze jméno uživatele, program nejprve předpokládá, že název je přezdívka, a odešle zprávu objektu definovanému tímto názvem. Pokud taková přezdívka neexistuje, program se pokusí odeslat zprávu do lokální fronty s tímto názvem.
- Pokud uživatel zadal jméno uživatele i jméno správce front, program odešle zprávu do poštovní fronty, která je definována těmito dvěma názvy.

Program nepřijímá prázdné zprávy a odstraňuje úvodní mezery z každého řádku textu zprávy.

Je-li volání MQPUT1 úspěšné, program zobrazí zprávu, která zobrazí jméno uživatele a jméno správce front, na které byla zpráva vložena. Je-li volání neúspěšné, program hledá speciálně pro kódy příčiny, které označují frontu nebo správce front neexistuje; jedná se o MQRC_UNKNOWN_OBJECT_NAME a MQRC_UNKNOWN_OBJECT_Q_MGR. Program zobrazí svou vlastní chybovou zprávu pro každou z těchto chyb; pro jiné chyby program zobrazí kódy dokončení a příčiny vrácené voláním.

z/OS *Program Nickname na systému z/OS*

Když uživatel definuje přezdívku, program přezdívek (CSQ4TVD5 pro TSO, CSQ4CVD5 pro CICS) vytvoří frontu, která má přezdívku jako část svého názvu.

Tento program to provede vložením zprávy do vstupní fronty příkazu systému. Zpráva obsahuje příkaz IBM MQ for z/OS DEFINE QALIAS nebo DEFINE QREMOTE. Typ fronty, který program vytváří, závisí na tom, jak uživatel vyplnil pole panelu Vytvořit přezdívku (CSQ4VDP5 pro TSO, VD5 pro CICS):

- Pokud uživatel zadal pouze jméno uživatele nebo jméno správce front je stejné jako jméno správce front, ke kterému je program Mail Manager připojen, program vytvoří alias frontu.
- Pokud uživatel zadal jméno uživatele i jméno správce front (a správce front není ten, ke kterému je správce pošty připojen), vytvoří program lokální definici vzdálené fronty. Program nekontroluje existenci fronty, na kterou je tento definice interpretována, nebo dokonce i existenci vzdáleného správce front.

Program také vytvoří dočasnou dynamickou frontu pro zpracování odpovědí ze vstupní fronty systémového příkazu.

Pokud správce front nemůže vytvořit frontu přezdívek z nějakého důvodu, který program očekává (například fronta již existuje), zobrazí se v programu vlastní chybová zpráva. Pokud správce front nemůže vytvořit frontu z důvodu, že program neočekává, program zobrazí až dvě z chybových zpráv, které jsou vráceny programu na příkazový server.

Poznámka: Pro každou přezdívku vytváří program pro přezdívku pouze alias fronty nebo lokální definici vzdálené fronty. Lokální fronty, do kterých jsou tyto názvy front vyřešeny, jsou vytvářeny pouze v případě, že je použito ID uživatele obsažené v přezdívce ke spuštění aplikace Správce pošty.

z/OS *Ukázka Credit Check na z/OS*

Ukázková aplikace Credit Check je sada programů, které ukazují, jak používat mnohé z funkcí poskytovaných produktem IBM MQ for z/OS. Ukazuje, jak mnoho aplikačních programů aplikace může předávat zprávy navzájem pomocí technik front zpráv.

Ukázku lze spustit jako samostatnou aplikaci produktu CICS . Chcete-li však demonstrovat, jak navrhnout aplikaci fronty zpráv, která používá prostředky poskytované jak v prostředí CICS , tak IMS , je jeden modul také dodáván jako dávkový program zpracování zpráv IMS . Toto rozšíření ukázky je popsáno v “Rozšíření IMS na ukázku Credit Check na z/OS” na stránce 1182.

Ukázku můžete také spustit ve více než jednom správci front a odesílat zprávy mezi každou instancí aplikace. Chcete-li to provést, prohlédněte si téma “Ukázka Credit Check s více správci front v systému z/OS” na stránce 1181

Programy produktu CICS jsou dodávány v jazycích C a COBOL. Jednotlivý program IMS se dodává pouze v C. Dodané datové sady jsou zobrazeny v Tabulka 171 na stránce 1152 a Tabulka 173 na stránce 1153.

Aplikace demonstruje metodu posouzení rizika, když se zákazníci banky žádají o půjčky. Aplikace zobrazuje, jak by banka mohla pracovat dvěma způsoby zpracování žádostí o půjčku:

- Při jednání přímo se zákazníkem chtějí bankovní pracovníci okamžitý přístup k informacím o účtu a úvěru-rizika.
- Při práci s psanými aplikacemi mohou bankovní zaměstnanci předkládat řadu žádostí o informace o účtech a úvěru-rizika a s odpověďmi se vypořádat později.

Finanční a bezpečnostní podrobnosti v žádosti byly zjednodušeny tak, aby techniky front zpráv byly jasné.

z/OS *Příprava a spuštění ukázky Credit Check na systému z/OS*

Chcete-li připravit a spustit ukázku Credit Check, proveďte následující kroky:

1. Vytvoření datové sady VSAM, která obsahuje informace o některých ukázkových účtech. Proveďte to úpravou a spuštěním souboru JCL dodaného v datové sadě CSQ4FILE.
2. Proveďte kroky uvedené v tématu “Příprava ukázkových aplikací pro prostředí CICS na systému z/OS” na stránce 1149. (Další kroky, které musíte provést, pokud chcete použít rozšíření produktu IMS na ukázku, jsou popsány v příručce “Rozšíření IMS na ukázku Credit Check na z/OS” na stránce 1182.)
3. Spustit monitor spouštěčů CKTI (dodávaný s IBM MQ for z/OS) do fronty CSQ4SAMP.INITIATION.QUEUEpomocí transakce CICS CKQC.
4. Chcete-li spustit aplikaci, spusťte svůj systém CICS a použijte transakci MVB1.
5. Vyberte volbu **Okamžité** nebo **Dávkové** dotazování z prvního panelu.

   Bezprostřední a dávkové dotazové panely jsou podobné; Obrázek 156 na stránce 1173 zobrazí panel Okamžitý dotaz.

```
CSQ4VB2       IBM MQ for z/OS Sample Programs

Credit Check - Immediate Inquiry

Specify details of the request, then press Enter.
Name . . . . . _____________________
Social security number ___ __ ____
Bank account name . . _______________________________
Account number . . . . __________
Amount requested . . . 012345
Response from CHECKING ACCOUNT for name : ____________________
Account information not found
Credit worthiness index - NOT KNOWN
..
..
..
..
..
..
..
..
..
MESSAGE LINE
F1=Help F3=Exit F5=Make another inquiry
```

Obrázek 156. Panel Okamžitý dotaz pro ukázkovou aplikaci kontroly kreditu

6. Zadejte číslo účtu a částku půjčky do příslušných polí. Informace o tom, jaké informace máte zadat do těchto polí, naleznete v příručce “Zadání informací do dotazových panelů” na stránce 1173 .

Zadání informací do dotazových panelů

Ukázková aplikace Kontrola kreditu kontroluje, zda data, která jste zadali do pole **Požadovaná částka** na panelech s dotazy, jsou ve formě celých čísel.

Zadáte-li jedno z následujících čísel účtů, aplikace najde odpovídající název účtu, průměrnou účetní bilanci a úvěrový index v datové sadě VSAM CSQ4BAQ:

- 2222222222
- 3111234329
- 3256478962
- 3333333333
- 3501676212
- 3696879656
- 4444444444
- 5555555555
- 6666666666
- 7777777777

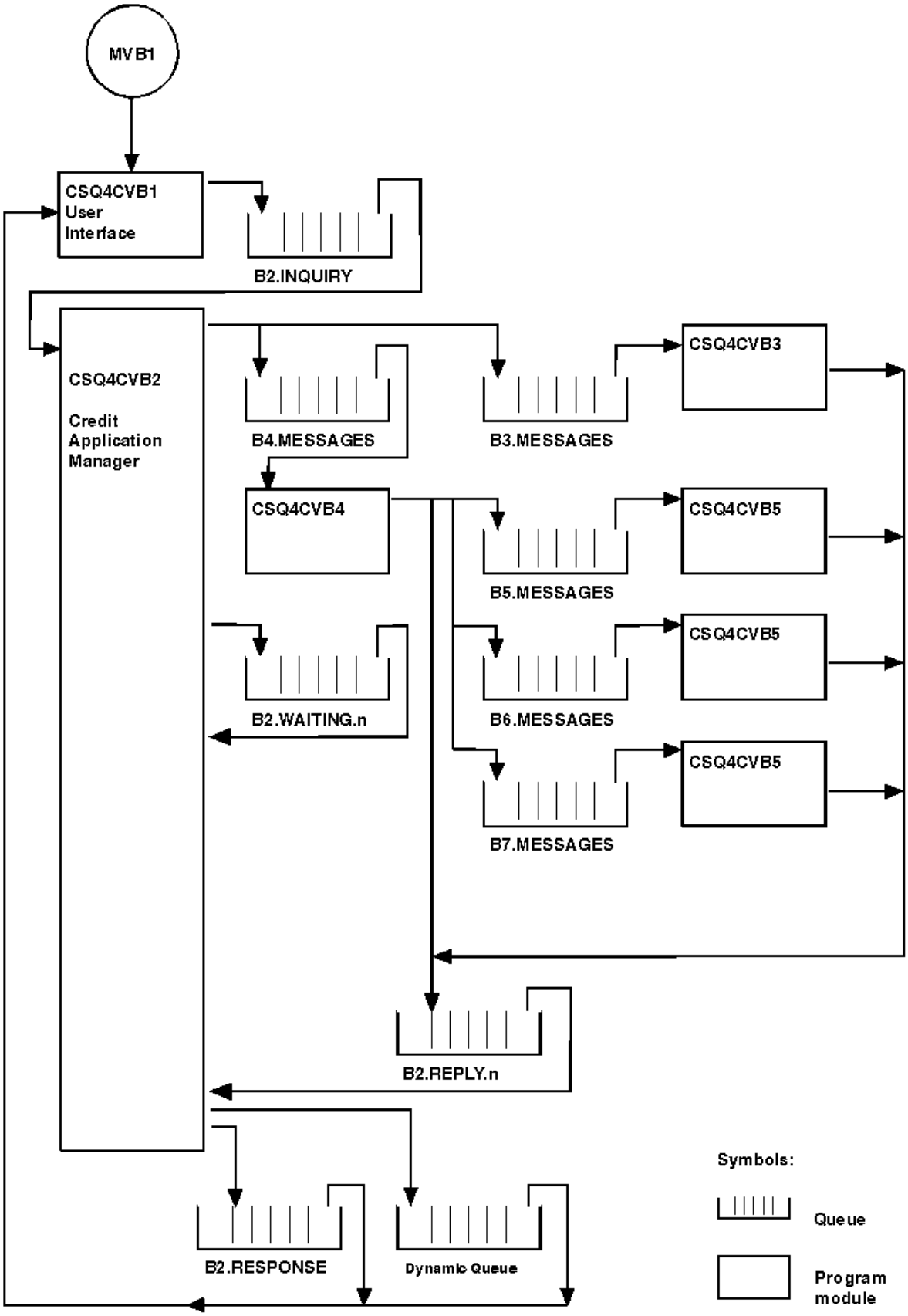
Do ostatních polí můžete zadat libovolné informace, nebo ne. Aplikace zachová všechny informace, které zadáte, a vrátí stejné informace v sestavách, které generuje.

z/OS *Návrh ukázky Credit Check na z/OS*

Tento oddíl popisuje návrh každého z programů, které tvoří ukázkovou aplikaci Kontrola kreditu.

Další informace o některých technikách, které byly zvažované během návrhu aplikace, viz “Aspekty návrhu pro vzorek kontroly kreditu v systému z/OS” na stránce 1179.

Obrázek 157 na stránce 1174 zobrazuje programy, které tvoří aplikaci, a také fronty, které tyto programy obsluhují. Na tomto obrázku byla předpona CSQ4SAMP vynechána ze všech názvů front, aby bylo možné lépe porozumět tomuto obrázku.



Obrázek 157. Programy a fronty pro ukázkovou aplikaci Kontrola kreditu (pouze programy v jazyce COBOL)

z/OS *Program uživatelského rozhraní (CSQ4CVB1) v systému z/OS*

Když spustíte transakci CICS MVB1v konverzačním režimu, spustí se tento program uživatelského rozhraní pro aplikaci.

Tento program umístí dotazové zprávy do fronty CSQ4SAMP.B2.INQUIRY a získává odpovědi na tyto dotazy z fronty pro odpovědi, které určuje, když provádí dotaz. Z uživatelského rozhraní můžete odesílat okamžité nebo dávkové dotazy:

- V případě okamžitých dotazů program vytvoří dočasnou dynamickou frontu, kterou používá jako frontu pro odpověď. To znamená, že každý dotaz má svou vlastní odpověď-do fronty.
- V případě dávkových dotazů obdrží program uživatelského rozhraní odpovědi z fronty CSQ4SAMP.B2.RESPONSE. Pro zjednodušení program získává odpovědi na všechny své dotazy z této jedné fronty pro odpovědi. Je snadné vidět, že banka může chtít použít samostatnou frontu pro odpověď pro každého uživatele MVB1tak, aby každá z nich mohla zobrazit odpovědi pouze na ty dotazy, které iniciovaly.

Důležité rozdíly mezi vlastnostmi zpráv použitých v aplikaci, když jsou v dávkovém režimu a v přímém režimu:

- Pro dávkovou práci mají zprávy nízkou prioritu, takže se zpracují po všech požadavcích na úvěr, které jsou zadány v okamžitém režimu. Zprávy jsou také trvalé, a proto jsou obnoveny v případě, že se má restartovat aplikace nebo správce front.
- Pro okamžité zpracování zpráv mají vysokou prioritu, takže jsou zpracovány před jakýmikoli požadavky na půjčky, které jsou zadány v dávkovém režimu. Zprávy také nejsou trvalé, a proto jsou zahozené v případě, že se má restartovat aplikace nebo správce front.

Ve všech případech se však vlastnosti zpráv žádostí o úvěr šíří po celé aplikaci. Takže například všechny zprávy, které jsou výsledkem požadavku vysoké priority, budou mít také vysokou prioritu.

z/OS *Správce aplikací úvěrů (CSQ4CVB2) v systému z/OS*

Program CAM (Credit Application Manager) provádí většinu zpracování pro aplikaci Credit Check.

Model CAM je spouštěn monitorem spouštěčů CKTI (dodávaný s IBM MQ for z/OS), když se událost triggeru vyskytne buď ve frontě CSQ4SAMP.B2.INQUIRY nebo frontu CSQ4SAMP.B2.REPLY. *n*, kde *n* je celé číslo, které identifikuje jednu ze sady front odpovědí. Zpráva spouštěče obsahuje data obsahující název fronty, v níž došlo k události spouštěče.

CAM používá fronty s názvy formulářů CSQ4SAMP.B2.WAITING.n ukládá informace o dotazech, které zpracovává. Fronty jsou pojmenovány tak, že jsou navzájem párovány s frontou odpovědí; například fronta CSQ4SAMP.B2.WAITING.3 obsahuje vstupní data pro konkrétní dotaz a frontu CSQ4SAMP.B2.REPLY.3 obsahuje sadu zpráv odpovědí (z programů, které dotazují databáze) všechny související s tímto dotazováním. Chcete-li pochopit příčiny tohoto návrhu, prohlédněte si téma “Samostatné dotazy a fronty odpovědí v prostředí CAM” na stránce 1179.

Logika spuštění

Pokud se událost triggeru vyskytne ve frontě CSQ4SAMP.B2.INQUIRY, modul CAM otevře frontu pro sdílený přístup. Poté se pokusí otevřít každou frontu odpovědí, dokud nebude nalezena volná jedna. Pokud nemůže najít volnou frontu odpovědí, CAM protokoluje tuto skutečnost a ukončuje normálně.

Pokud se událost triggeru vyskytne ve frontě CSQ4SAMP.B2.REPLY.n, CAM otevře frontu pro výlučný přístup. Pokud návratový kód ohlásí, že objekt se již používá, platnost údržby společných prostor je normálně ukončena. Dojde-li k jiné chybě, obsah protokolu CAM zaznamená chybu a skončí. CAM otevře odpovídající frontu čekání a frontu dotazů, pak začne přijímat a zpracovávat zprávy. Z fronty čekání napraví modul CAM podrobnosti o částečně dokončeném šetření.

Pro zjednodušení v této ukázce se v programu nacházejí názvy front používaných v programu. V obchodním prostředí by názvy front pravděpodobně byly zadrženy v souboru, k němuž má program přístup.

Získání zprávy z fronty dotazů

Modul CAM se nejprve pokusí o získání zprávy z fronty dotazů pomocí volání MQGET s volbou MQGMO_SET_SIGNAL. Je-li zpráva okamžitě k dispozici, je zpráva zpracována; pokud není k dispozici žádná zpráva, je nastaven signál.

Modul CAM se poté pokusí o získání zprávy z fronty odpovědí znovu s použitím volání MQGET se stejnou volbou. Je-li zpráva okamžitě k dispozici, je zpráva zpracována; v opačném případě je nastaven signál.

Když jsou oba signály nastaveny, program čeká, dokud se neodešle jeden z signálů. Je-li odeslán signál k označení, že je zpráva k dispozici, zpráva se načte a zpracuje. Pokud dojde k vypršení platnosti signálu nebo ukončení činnosti správce front, program se ukončí.

Zpracování zprávy načtené pomocí modulu CAM

Zpráva načtená pomocí modulu CAM může být jedním ze čtyř typů:

- Dotazová zpráva
- Zpráva odpovědi
- Zpráva o šíření
- Neočekávaná nebo nežádoucí zpráva

CAM zpracovává tyto zprávy, jak je popsáno v tématu “Zpracování zprávy načtené pomocí modulu CAM v systému z/OS” na stránce 1176.

Odesílání odpovědi

Když CAM přijme všechny odpovědi, které očekává pro dotaz, zpracuje odpovědi a vytvoří jednu zprávu odezvy. Zkonsoliduje do jedné zprávy všechna data ze všech zpráv odpovědí, které mají stejný `CorrelId`. Tato odezva je vložena do fronty pro odpovědi určené v původním požadavku na půjčku. Zpráva odpovědi se umístí do stejné jednotky práce, která obsahuje načtení konečné zprávy odpovědi. Tím se zjednodušuje obnova tím, že se ujistíte, že ve frontě CSQ4SAMP.B2.WAITING.n.

Obnova částečně dokončeného šetření

CAM kopíruje do fronty CSQ4SAMP.B2.WAITING.n . Všechny zprávy, které přijímá. Nastavuje pole deskriptoru zpráv takto:

- *`Priority`* je určen typem zprávy:
  - Pro zprávy požadavku, priorita = 3
  - Pro datagramy, priorita = 2
  - Pro zprávy odpovědí, priorita = 1
- *`CorrelId`* je nastaven na *`MsgId`* zprávy o žádosti o úvěr.
- Ostatní pole MQMD se zkopírují ze zpráv přijaté zprávy

Když bylo dokončeno vyšetřování, zprávy pro specifický dotaz se odstraní z čekací fronty během zpracování odpovědi. Proto vždy čekající fronta obsahuje všechny zprávy vztahující se k probíhajícím šet%ení. Tyto zprávy se používají k obnovení podrobností o dotazech na průběh, pokud se má program restartovat. Různé priority jsou nastaveny tak, že dotazové zprávy jsou obnoveny před propagací nebo odpověďmi zpráv.

z/OS *Zpracování zprávy načtené pomocí modulu CAM v systému z/OS*

Zpráva načtená správcem CAM (Credit Application Manager) může být jedním ze čtyř typů. Způsob, jakým modul CAM zpracovává zprávu, závisí na jeho typu.

Zpráva načtená pomocí modulu CAM může být jedním ze čtyř typů:

- Dotazová zpráva
- Zpráva odpovědi

- Zpráva o šíření
- Neočekávaná nebo nežádoucí zpráva

CAM tyto zprávy zpracovává následujícím způsobem:

Dotazová zpráva
Dotazové zprávy pocházejí z programu uživatelského rozhraní. Vytvoří dotazovou zprávu pro každý požadavek na půjčku.

Pro všechny žádosti o úvěr si model CAM vyžádá průměrnou zůstatek na účtu kontroly zákazníka. To se provede tak, že se odešle zpráva požadavku na alias frontu CSQ4SAMP.B2.OUTPUT.ALIAS. Tento název fronty se interpretuje jako fronta CSQ4SAMP.B3.MESSAGES, který je zpracován programem checking-account, CSQ4CVB3. Když modul CAM vloží zprávu do této alias fronty, uvádí příslušnou hodnotu CSQ4SAMP.B2.REPLY.n fronta pro odpověď na frontu. Zde se používá alias fronta, takže program CSQ4CVB3 může být snadno nahrazen jiným programem, který zpracovává základní frontu jiného jména. Chcete-li tuto operaci provést, předefinujte frontu aliasů tak, aby se její název interpretoval jako nová fronta. Také můžete přiřadit odlišné přístupové oprávnění k frontě aliasů a k základní frontě.

Pokud uživatel požádá o půjčku, která je větší než 10000 jednotek, modul CAM zahájí také kontrolu nad ostatními databázemi. To se provede tak, že se do fronty odešle zpráva CSQ4SAMP.B4.MESSAGES, který je zpracován distribučním programem, CSQ4CVB4. Proces obsluhující tuto frontu šíří zprávu do front obsluhovaných programy, které mají přístup k dalším záznamům, jako jsou např. historie kreditní karty, spořicí účty a hypoteční platby. Data z těchto programů jsou vrácena do fronty pro odpovědi zadané v operaci put. Kromě toho se do fronty pro odpověď odešle zpráva o šíření, která určí, kolik zpráv o šíření bylo odesláno.

V obchodním prostředí by distribuční program pravděpodobně přeformátoval data poskytnutá tak, aby odpovídala formátu požadovanému každým z ostatních typů bankovního účtu.

Kterákoliv z uvedených front může být ve vzdáleném systému.

Pro každou dotazovou zprávu spustí CAM záznam v tabulce záznamů Inquirese Inquiry Record Table (IRT). Tento záznam obsahuje:

- `MsgId` dotazové zprávy
- V poli ReplyExp očekávaný počet odpovědí (rovnající se počtu odeslaných zpráv)
- V poli ReplyRec počet přijatých odpovědí (od nuly v této fázi)
- V poli PropsOut informaci o tom, zda se očekává zpráva o šíření

CAM zkopíruje dotazovou zprávu do fronty čekání pomocí:

- `Priority` nastaveno na 3
- `CorrelId` nastaven na `MsgId` dotazové zprávy
- Ostatní pole deskriptoru zpráv, která jsou nastavena na ty z dotazové zprávy

Zpráva propagace
Zpráva šíření obsahuje počet front, do kterých distribuční program předal dotaz. Zpráva se zpracovává následujícím způsobem:

1. Přidejte do pole ReplyExp příslušného záznamu v modelu IRT, počet odeslaných zpráv. Tyto informace jsou ve zprávě.
2. Increment by 1 pole ReplyRec záznamu v modelu IRT.
3. Snížit o 1 pole PropsOut záznamu v rámci projektu IRT.
4. Zkopírujte zprávu do fronty čekání. CAM nastaví `Priority` na 2 a další pole deskriptoru zpráv na hodnoty zprávy šíření.

Odpověď na zprávu
Zpráva odpovědi obsahuje odpověď na jeden z požadavků na program checking-account nebo na jeden z programů agentur-query. Zprávy odpovědí jsou zpracovány následujícím způsobem:

1. Increment by 1 pole ReplyRec záznamu v modelu IRT.

2. Zkopírujte zprávu do fronty čekání s parametrem `Priority` nastaveným na hodnotu 1 a ostatními poli deskriptoru zpráv nastaveným na hodnotu uvedenou ve zprávě s odpovědí.
3. Je-li položka ReplyRec = ReplyExpa PropsOut = 0, nastavte příznak MsgComplete .

Další zprávy

Aplikace neočekává další zprávy. Aplikace však může přijímat zprávy vysílané systémem nebo přijímat zprávy s neznámým serverem `CorrelIds`.

CAM vkládá tyto zprávy do fronty CSQ4SAMP.DEAD.QUEUE, kde je možné je vyšetřit. Pokud tato operace vložení selže, zpráva se ztratí a program pokračuje. Další informace o návrhu této části programu najdete v tématu "Jak ukázka zpracovává neočekávané zprávy" na stránce 1180.

z/OS *Kontrola-účtovací program (CSQ4CVB3) na z/OS*

Program checking-account je spuštěn událostí triggeru ve frontě CSQ4SAMP.B3.MESSAGES. Po otevření fronty získá tento program zprávu z fronty pomocí volání MQGET s volbou wait a s intervalem čekání nastaveným na 30 sekund.

Program prohledá datovou sadu VSAM CSQ4BAQ pro číslo účtu ve zprávě požadavku na půjčku. Získává odpovídající název účtu, průměrnou bilanci a index letové způsobilosti, nebo bere na vědomí, že číslo účtu není v datové sadě.

Program poté vloží zprávu s odpovědí (pomocí volání MQPUT1) do fronty pro odpovědi uvedené ve zprávě o požadavku na půjčku. Pro tuto zprávu odpovědi program:

- Zkopíruje `CorrelId` zprávy požadavku na půjčku.
- Používá volbu MQPMO_PASS_IDENTITY_CONTEXT

Program pokračuje ve získávání zpráv z fronty, dokud nevyprší interval čekání.

z/OS *Distribuční program (CSQ4CVB4) v systému z/OS*

Distribuovací program je spuštěn událostí triggeru ve frontě CSQ4SAMP.B4.MESSAGES.

Chcete-li simulovat rozdělení žádosti o úvěr na jiné agentury, které mají přístup k záznamům jako jsou historie kreditní karty, spořicí účty a hypoteční platby, program umístí kopii stejné zprávy na všechny fronty v seznamu názvů CSQ4SAMP.B4.NAMELIST. K dispozici jsou tři z těchto front s názvy ve formátu CSQ4SAMP.B *n*.MESSAGES, kde *n* je 5, 6, nebo 7. V obchodní aplikaci mohou být agentury v oddělených umístěních, takže tyto fronty mohou být vzdálené fronty. Chcete-li upravit ukázkovou aplikaci, abyste ji viděli, prohlédněte si téma "Ukázka Credit Check s více správci front v systému z/OS" na stránce 1181.

Distribuční program provádí následující kroky:

1. Ze seznamu názvů získáte názvy front, které má program používat. Tento program provádí toto volání pomocí volání MQINQ k dotazu na atributy objektu seznamu názvů.
2. Otevře tyto fronty a také CSQ4SAMP.B4.MESSAGES.
3. Provede následující smyčku, dokud ve frontě nejsou žádné další zprávy CSQ4SAMP.B4.MESSAGES:
   a. Získejte zprávu pomocí volání MQGET s volbou wait a s intervalem čekání nastaveným na 30 sekund.
   b. Vložte zprávu do každé fronty uvedené v seznamu názvů a zadejte název odpovídající proměnné CSQ4SAMP.B2.REPLY.n fronta pro odpověď na frontu. Program zkopíruje *`CorrelId`* zprávy o žádosti o úvěr na tyto kopírovací zprávy a použije volbu MQPMO_PASS_IDENTITY_CONTEXT na volání MQPUT.
   c. Odešlete zprávu datagramu do fronty CSQ4SAMP.B2.REPLY.n zobrazí počet zpráv, které byly úspěšně vloženy.
   d. Deklarujte synchronizační bod.

z/OS *Program dotazovacích programů (CSQ4CVB5/CSQ4CCB5) v systému z/OS*

Program-dotaz-dotaz se dodává jako program v jazyku COBOL i do programu C. Oba programy mají stejný návrh. To ukazuje, že programy různých typů mohou být v aplikaci IBM MQ snadno koexistovat a že programové moduly, které tvoří takovou aplikaci, lze snadno nahradit.

Instance programu je spouštěna událostí triggeru na kterékoliv z těchto front:

- Pro program v jazyce COBOL (CSQ4CVB5):
  - CSQ4SAMP.B5.MESSAGES
  - CSQ4SAMP.B6.MESSAGES
  - CSQ4SAMP.B7.MESSAGES
- Pro program v jazyce C (CSQ4CCB5), fronta CSQ4SAMP.B8.MESSAGES

Poznámka: Chcete-li použít program v jazyce C, je třeba změnit definici seznamu názvů CSQ4SAMP.B4.NAMELIST nahradí frontu CSQ4SAMP.B7.MESSAGES s CSQ4SAMP.B8.MESSAGES. Chcete-li to provést, můžete použít libovolnou z následujících možností:

- Ovládací panely a ovládací panely produktu IBM MQ for z/OS
- Příkaz ALTER NAMELIST
- Obslužný program CSQUTIL

Po otevření příslušné fronty tento program získá zprávu z fronty pomocí volání MQGET s volbou wait a s intervalem čekání nastaveným na 30 sekund.

Program simuluje vyhledávání v databázi agentury prohledáním datové sady VSAM CSQ4BAQ pro číslo účtu, které bylo předáno ve zprávě požadavku na půjčku. Poté sestaví odpověď, která obsahuje název fronty, kterou obsluhuje, a index úvěruschopnosti. Pro zjednodušení zpracování je index úvěruschopnosti vybrán náhodně.

Při vkládání zprávy s odpovědí program používá volání MQPUT1 a:

- Zkopíruje `CorrelId` zprávy požadavku na půjčku.
- Používá volbu MQPMO_PASS_IDENTITY_CONTEXT

Program pošle zprávu odpovědi do fronty odpovědí uvedené ve zprávě požadavku na úvěr. (Název správce front, který je vlastníkem fronty pro odpovědi, je také určen ve zprávě požadavku na půjčku.)

z/OS *Aspekty návrhu pro vzorek kontroly kreditu v systému z/OS*

Aspekty návrhu pro ukázku Credit Check.

Toto téma obsahuje informace o následujících tématech:

Samostatné dotazy a fronty odpovědí v prostředí CAM

Aplikace může používat jednu frontu pro oba dotazy a odpovědi, ale byla navržena tak, aby používala oddělené fronty, a to z následujících důvodů:

- Když program zpracovává maximální počet dotazů, další dotazy mohou být ponechány na frontě. Je-li použita jediná fronta, je třeba ji z fronty odstranit a uložit jinam.
- Ostatní instance CAM by mohly být spuštěny automaticky pro obsloužení stejné dotazové fronty, pokud by přenos zpráv byl natolik vysoký, aby mohl být zatykač. Program však musí sledovat probíhající dotazy

a musí to udělat, musí vrátit všechny odpovědi na dotazy, které inicioval. Je-li použita pouze jedna fronta, program bude muset projít zprávy, aby zjistil, zda byli pro tento program nebo pro jiný. Tím by byla operace mnohem méně efektivní.

Aplikace může podporovat více aplikací CAMs a efektivně provádět probíhající dotazy pomocí párovaných odpovědí a čekajících front.

- Program může efektivně čekat na více frontách pomocí signalizace.

Jak ukázka zpracovává chyby

Program uživatelského rozhraní zpracovává chyby tím, že je nahlásí přímo uživateli.

Ostatní programy nemají uživatelská rozhraní, takže musí ošetřit chyby jinými způsoby. Také v mnoha situacích (například, pokud se volání MQGET nezdaří), tyto jiné programy neznají identitu uživatele aplikace.

Ostatní programy umístí chybové zprávy do dočasné paměťové fronty CICS s názvem CSQ4SAMP. Tuto frontu můžete procházet pomocí transakce CEBR dodaná produktem CICS. Programy také zapisují chybové zprávy do protokolu CICS CSML.

Jak ukázka zpracovává neočekávané zprávy

Když navrhnete aplikaci zařazování do fronty zpráv, musíte se rozhodnout, jak zpracovat zprávy, které dorazí do fronty neočekávaně.

K dispozici jsou dvě základní volby:

- Aplikace neprovede žádnou další práci, dokud nezpracovala neočekávanou zprávu. To pravděpodobně znamená, že aplikace informuje operátora, ukončí se a ujistí se, že se automaticky nerestartuje (může to provést nastavením spouštění). Tato volba znamená, že veškeré zpracování pro aplikaci může být zastaveno jednou neočekávanou zprávou a zásah operátora je nutný pro restartování aplikace.
- Aplikace odstraní zprávu z fronty, kterou obsluhuje, vloží ji do jiného umístění a pokračuje ve zpracování. Nejlepším místem pro vložení této zprávy je fronta nedoručených zpráv systému.

Vyberete-li druhou volbu, postupujte takto:

- Operátor nebo jiný program by měl zkontrolovat zprávy, které jsou vloženy do fronty nedoručených zpráv, aby bylo možné zjistit, odkud přicházejí zprávy.
- Pokud ji nelze umístit do fronty nedoručených zpráv, dojde ke ztrátě neočekávané zprávy.
- Dlouhá neočekávaná zpráva je oříznuta, pokud je delší než limit pro zprávy ve frontě nedoručených zpráv nebo delší než velikost vyrovnávací paměti v programu.

Aby se zajistilo, že aplikace hladce zpracovává všechny dotazy s minimálním účinkem od vnějších aktivit, použije ukázková aplikace Credit Check druhou volbu. Chcete-li nechat ukázku oddělit od jiných aplikací, které používají stejného správce front, ukázka Credit Check nevyužívá frontu nedoručených zpráv systému; místo toho používá vlastní frontu zablokovaných dopisů. Tato fronta má název CSQ4SAMP.DEAD.QUEUE. Ukázka zkrátí všechny zprávy, které jsou delší než velikost vyrovnávací paměti poskytnuté pro ukázkové programy. K procházení zpráv v této frontě můžete použít ukázkovou aplikaci Procházet nebo pomocí ukázkové aplikace Tisková zpráva vytisknout zprávy spolu s jejich deskriptory zpráv.

Pokud však rozšíření ukázky rozšíříte na více než jednoho správce front, lze správce front uložit do fronty zablokovaných zpráv neočekávané zprávy nebo zprávy, které nelze doručit.

Jak ukázka používá synchronizační body

Programy v ukázkové aplikaci pro kontrolu kreditů deklarují synchronizační body, aby bylo zajištěno, že:

- V odpovědi na každou očekávanou zprávu se odešle pouze jedna zpráva odpovědi.
- Vícenásobné kopie neočekávaných zpráv se nikdy nevloží do fronty dead-letter ukázky

- CAM může obnovit stav všech částečně dokončených dotazů tím, že bude z fronty čekajících zpráv získávat trvalé zprávy.

K dosažení tohoto cíle je použita jediná jednotka práce, která pokryje získání zprávy, zpracování této zprávy a všechny následné operace vložení.

Jak ukázka používá informace o kontextu zprávy

Když program uživatelského rozhraní (CSQ4CVB1) odešle zprávy, použije volbu MQPMO_DEFAULT_CONTEXT. To znamená, že správce front generuje informace o kontextu identity a původu. Správce front získá tyto informace z transakce, která spustila program (MVB1), a od ID uživatele, který spustil transakci.

Když CAM odesílá dotazové zprávy, používá volbu MQPMO_PASS_IDENTITY_CONTEXT. To znamená, že informace o kontextu identity odesílaného zprávy se kopírují z kontextu identity původní dotazové zprávy. S touto volbou jsou informace o kontextu původu generovány správcem front.

Když modul CAM odešle zprávy s odpovědí, používá volbu MQPMO_ALTERNATE_USER_AUTHORITY. To způsobí, že správce front použije alternativní ID uživatele pro kontrolu zabezpečení, když prostředí CAM otevře frontu pro odpověď na frontu. CAM používá ID uživatele, který předkladatel původní dotazové zprávy zadal. To znamená, že uživatelé mají povoleno zobrazit odpovědi pouze na ty dotazy, které pocházejí. Alternativní ID uživatele se získá z informací o kontextu identity v deskriptoru zprávy původní dotazové zprávy.

Když dotazovací programy (CSQ4CVB3/4/5) odesílají zprávy s odpovědí, používají volbu MQPMO_PASS_IDENTITY_CONTEXT. To znamená, že informace o kontextu identity odesílaného zprávy se kopírují z kontextu identity původní dotazové zprávy. S touto volbou jsou informace o kontextu původu generovány správcem front.

Poznámka: ID uživatele přidružené k transakcím MVB3/4/5 vyžaduje přístup k souboru B2.REPLY.n front. Tyto ID uživatele nemusí být stejné jako ty, které jsou přidruženy k zpracovávenému požadavku. Pro získání této možné bezpečnostní expozice by dotazovací programy mohly při vkládání odpovědí použít volbu MQPMO_ALTERNATE_USER_AUTHORITY. To by znamenalo, že každý jednotlivý uživatel MVB1 potřebuje oprávnění k otevření B2.REPLY.n front.

Použití identifikátorů zpráv a korelace v prostředí CAM

Aplikace musí monitorovat průběh všech aktuálních dotazů, které zpracovává, a to v libovolném okamžiku. Chcete-li to provést, použije jedinečný identifikátor zprávy každé zprávy o požadavku na půjčku k přidružení všech informací, které má o každém dotazu.

CAM zkopíruje `MsgId` z dotazové zprávy do `CorrelId` všech zpráv požadavku, které odešle pro tento dotaz. Ostatní programy ve vzorku (CSQ4CVB3 -5) zkopírujte `CorrelId` každé zprávy, kterou obdrží do `CorrelId` své odpovědi na zprávu.

z/OS *Ukázka Credit Check s více správci front v systému z/OS*

Ukázkovou aplikaci Kontrola kreditu můžete použít k demonstraci distribuovaných front prostřednictvím instalace ukázky ve dvou správcích front a systémech CICS (s každým správcem front připojeným k jinému systému CICS).

Když je nainstalován ukázkový program a monitor spouštěčů (CKTI) je spuštěný na každém systému, musíte:

1. Nastavte komunikační spojení mezi dvěma správci front. Informace o tom, jak to provést, najdete v tématu Konfigurace distribuovaných front.
2. V jednom správci front vytvořte lokální definici pro každou vzdálenou frontu (v jiném správci front), kterou chcete použít. Tyto fronty mohou být libovolné z CSQ4SAMP.B *n*.MESSAGES, kde *n* je 3, 5, 6, nebo 7. (Jedná se o fronty, které jsou obsluhovány programem checking-account a programem-query-query.) Další informace o tom, jak to provést, najdete v tématu DEFINE QREMOTE a DEFINE queues.
3. Změňte definici seznamu názvů (CSQ4SAMP.B4.NAMELIST) tak, aby obsahoval názvy vzdálených front, které chcete použít. Další informace o tom, jak to provést, najdete v tématu DEFINE NAMELIST.

z/OS *Rozšíření IMS na ukázku Credit Check na z/OS*

Verze programu checking-account se dodává jako program dávkového zpracování zpráv produktu IMS (BMP). Je napsán v jazyce C.

Program provádí stejnou funkci jako verze produktu CICS , kromě toho, že má získat informace o účtu, program čte databázi IMS místo souboru VSAM. Pokud nahradíte CICS verzi programu checking-account s verzí produktu IMS , nezobrazí se žádný rozdíl v metodě použití aplikace.

Chcete-li připravit a spustit verzi produktu IMS , musíte:

1. Postupujte podle pokynů v části "Příprava a spuštění ukázky Credit Check na systému z/OS" na stránce 1172.
2. Postupujte podle pokynů v části "Příprava ukázkové aplikace pro prostředí produktu IMS v systému z/OS" na stránce 1153.
3. Upravte definici alias fronty CSQ4SAMP.B2.OUTPUT.ALIAS pro vyřešení na frontu CSQ4SAMP.B3.IMS.MESSAGES (místo CSQ4SAMP.B3.MESSAGES). Chcete-li to provést, můžete použít jednu z následujících možností:
   - Ovládací panely a ovládací panely produktu IBM MQ for z/OS
   - Příkaz ALTER QALIAS .

Jiný způsob použití programu pro kontrolu účtu IMS je, že má sloužit jako jedna z front, která přijímá zprávy z distribučního programu. V dodané podobě ukázkové aplikace pro kontrolu kreditu jsou tři z těchto front (B5/6/7.MESSAGES), vše obsluhované programem-query program. Tento program prohledá datovou sadu VSAM. Chcete-li porovnat použití datové sady VSAM a databáze IMS , můžete místo toho zajistit, aby program pro kontrolu kontrolního účtu produktu IMS sloužil jedné z těchto front. Chcete-li tak učinit, je třeba upravit definici seznamu názvů CSQ4SAMP.B4.NAMELIST nahradí jednu z CSQ4SAMP.B *n*.Fronty MESSAGES s CSQ4SAMP.B3.IMS.Fronta MESSAGES. Můžete použít jednu z následujících možností:

- Ovládací panely a ovládací panely produktu IBM MQ for z/OS
- Příkaz ALTER NAMELIST .

Poté můžete spustit ukázku z transakce CICS MVB1. Uživatel nevidí žádný rozdíl v operaci nebo odezvě. BMP IMS se zastaví buď po přijetí zprávy o zastavení, nebo poté, co bude neaktivní po dobu pěti minut.

Návrh programu IMS checking-account (CSQ4ICB3)

Tento program běží jako BMP. Spusťte program pomocí JCL před tím, než jsou do něj odeslány jakékoli zprávy produktu IBM MQ .

Program prohledá databázi IMS pro číslo účtu ve zprávách žádosti o úvěr. Získává odpovídající název účtu, průměrnou bilanci a index letové způsobilosti.

Program odešle výsledky hledání databáze do fronty pro odpověď pojmenované ve zpracovávané zprávě produktu IBM MQ . Vrácená zpráva připojí typ účtu a výsledky vyhledávání na přijatou zprávu, takže transakce může potvrdit, že je zpracováván správný dotaz. Zpráva se nachází ve formě tří 79-znakových skupin:

```
'Response from CHECKING ACCOUNT for name : JONES J B'
'   Opened 870530, 3-month average balance = 000012.57'
'   Credit worthiness index - BBB'
```

Je-li program spuštěn jako soubor BMP orientovaný na zprávy, program odvodí frontu zpráv produktu IMS a poté čte zprávy z fronty produktu IBM MQ for z/OS a zpracovává je. Z fronty zpráv IMS se nepřijímají žádné informace. Program se po každém kontrolním bodu znovu připojí ke správci front, protože manipulátory byly uzavřeny.

Při spuštění v dávce BMP orientované na dávky je program po každém kontrolním bodu stále připojen ke správci front, protože manipulátory nejsou zavřeny.

z/OS *Ukázka obslužné rutiny zpráv v systému z/OS*

Ukázková aplikace TSO pro obslužnou rutinu zpráv vám umožňuje procházet, předávat a odstraňovat zprávy ve frontě. Ukázka je k dispozici v jazycích C a COBOL.

Příprava a spuštění ukázky

Postupujte takto:

1. Připravte ukázku tak, jak je popsáno v části “Příprava ukázkových aplikací pro prostředí TSO v systému z/OS” na stránce 1147.
2. Přizpůsobte rozhraní CLIST (CSQ4RCH1) pro ukázku, abyste definovali umístění panelů, umístění souboru zpráv a umístění zaváděcích modulů.

Můžete použít CLIST CSQ4RCH1 ke spuštění jak C, tak i COBOL verze vzorku. Dodaná verze CSQ4RCH1 spustí verzi jazyka C a obsahuje pokyny k přizpůsobení, které je nezbytné pro verzi jazyka COBOL.

Poznámka:

1. K dispozici nejsou žádné ukázkové definice front s ukázkou.
2. VS COBOL II nepodporuje souběžné zpracování více úloh s ISPF, takže nepoužívejte ukázkovou aplikaci pro obslužnou rutinu zpráv na obou stranách rozdělené obrazovky. Pokud ano, výsledky jsou nepředvídatelné.

z/OS *Použití ukázky obslužné rutiny zpráv v systému z/OS*

Poté, co jste nainstalovali ukázku a vyvolali ji z upraveného CLIST CSQ4RCH1, zobrazí se obrazovka zobrazovaná v produktu Obrázek 158 na stránce 1183 .

```
----------------------- IBM MQ for z/OS -- Samples ------------------------
COMMAND ===>
User Id : JOHNJ



Enter information. Press ENTER :

Queue Manager Name  : ________________________________________________ :

Queue Name      : ________________________________________________ :









F1=HELP   F2=SPLIT   F3=END    F4=RETURN  F5=RFIND   F6=RCHANGE
F7=UP    F8=DOWN   F9=SWAP   F10=LEFT   F11=RIGHT  F12=RETRIEVE
```

Obrázek 158. Výchozí obrazovka pro ukázku obslužné rutiny zpráv

Zadejte název správce front a název fronty, který má být zobrazen (rozlišuje velikost písmen) a zobrazí se obrazovka se seznamem zpráv (viz Obrázek 159 na stránce 1184).

```
---------------------- IBM MQ for z/OS -- Samples ------- Row 1 to 4 of 4
COMMAND ==>

Queue Manager  : VM03                        :
Queue      : MQEI.IMS.BRIDGE.QUEUE               :

Message number  01 of 04

Msg  Put Date  Put Time Format   User   Put Application
No  MM/DD/YYYY HH:MM:SS Name  Identifier  Type   Name
01  10/16/1998 13:51:19 MQIMS  NTSFV02   00000002 NTSFV02A
02  10/16/1998 13:55:45 MQIMS  JOHNJ    00000011 EDIT\CLASSES\BIN\PROGTS
03  10/16/1998 13:54:01 MQIMS  NTSFV02   00000002 NTSFV02B
04  10/16/1998 13:57:22 MQIMS  johnj    00000011 EDIT\CLASSES\BIN\PROGTS
****************************** Bottom of data ********************************
```

Obrázek 159. Obrazovka se seznamem zpráv pro ukázku obslužné rutiny zpráv

Na této obrazovce se zobrazí prvních 99 zpráv ve frontě a pro každý z nich jsou uvedena následující pole:

Číslo zprávy
Číslo zprávy.

Datum vložení MM/DD/RRRR
Datum, kdy byla zpráva vložena do fronty (GMT)

Čas vložení HH:MM: SS
Čas, kdy byla zpráva vložena do fronty (GMT)

Název formátu
MQMD.Format

Identifikátor uživatele
MQMD.UserIdentifier , pole

Typ vkládající aplikace
MQMD.PutApplType

Název vkládající aplikace
MQMD.PutApplName

Zobrazí se také celkový počet zpráv ve frontě.

Na této obrazovce může být vybrána zpráva, nikoli pozicí kurzoru, a poté se zobrazí. Příklad viz Obrázek 160 na stránce 1185.

```
---------------------- IBM MQ for z/OS -- Samples ----- Row 1 to 35 of 35
COMMAND ==>

Queue Manager  : VM03                       :
Queue      : MQEI.IMS.BRIDGE.QUEUE              :
Forward to Q Mgr : VM03                        :
Forward to Queue : QL.TEST.ISCRES1                  :

Action : _ :  (D)elete (F)orward

Message Content :
--------------------------------------------------------------------------------
Message Descriptor
StrucId     : `MD `
Version     : 000000001
Report      : 000000000
MsgType     : 000000001
Expiry      : -00000001
Feedback     : 000000000
Encoding     : 000000785
CodedCharSetId  : 000000500
Format      : `MQIMS  `
Priority     : 000000000
Persistence    : 000000001
MsgId      : `C3E2D840E5D4F0F34040404040404040AF6B30F0A89B7605`X
CorrelId     : `00000000000000000000000000000000000000000000000000000000`X
BackoutCount    : 000000000
ReplyToQ     : `QL.TEST.ISCRES1                  `
ReplyToQMgr    : `VM03                        `
UserIdentifier  : `NTSFV02   `
AccountingToken :
`06F2F5F5F3F0F1000000000000000000000000000000000000000000000000000000`X
ApplIdentityData : `                            `
PutApplType    : 000000002
PutApplName    : `NTSFV02A          `
PutDate     : `19971016`
PutTime     : `13511903`
ApplOriginData  : `   `

Message Buffer :  108 byte(s)
00000000 : C9C9 C840 0000 0001 0000 0054 0000 0311 `IIH .............`
00000010 : 0000 0000 4040 4040 4040 4040 0000 0000 `....    ....`
00000020 : 4040 4040 4040 4040 4040 4040 4040 4040 `                `
00000030 : 4040 4040 4040 4040 4040 4040 4040 4040 `                `
00000040 : 0000 0000 0000 0000 0000 0000 0000 0000 `................`
00000050 : 40F1 C300 0018 0000 C9C1 D7D4 C4C9 F2F8 ` 1C.....IAPMDI28`
00000060 : 40C8 C5D3 D3D6 40E6 D6D9 D3C4      ` HELLO WORLD  `
****************************** Bottom of data ********************************
```

Obrázek 160. Zvolená zpráva se zobrazí

Jakmile se zpráva zobrazí, může být odstraněna, ponechána ve frontě nebo předána do jiné fronty. Pole `Forward to Q Mgr` a `Forward to Queue` jsou inicializována hodnotami z MQMD, lze je změnit před předáním zprávy.

Návrh ukázky umožňuje vybrat a zobrazit pouze zprávy s jedinečnými kombinacemi `MsgId / CorrelId` , protože zpráva je načtena s použitím `MsgId` a `CorrelId` jako klíče. Pokud klíč není jedinečný, ukázka nemůže načíst zvolenou zprávu s jistotou.

Poznámka: Použijete-li vzorek SCSQCLST (CSQ4RCH1) k procházení zpráv, způsobí každé vyvolání počet vrácených zpráv, které se mají zvýšit. Chcete-li změnit chování této ukázky, zkopírujte ukázku a upravte obsah podle potřeby. Měli byste si být vědomi toho, že ostatní aplikace, které se spoléhají na tento počet odvolání, mohou být ovlivněny tímto rostoucím počtem.

z/OS *Návrh ukázky obslužné rutiny zpráv v systému z/OS*

Toto téma popisuje návrh každého z programů, které tvoří ukázkovou aplikaci Popisovač zpráv.

Program pro ověření objektů

To vyžaduje platnou frontu a název správce front.

Pokud není zadán název správce front, bude použit výchozí správce front, je-li k dispozici. Lze použít pouze lokální fronty; je zadán příkaz MQINQ, který kontroluje, zda je fronta typu fronta hlášena, a pokud fronta není lokální, je hlášena chyba. Pokud není fronta úspěšně otevřena nebo je ve frontě zablokováno volání MQGET, jsou vráceny chybové zprávy označující návratový kód CompCode a Reason return code.

Program v seznamu zpráv

Zobrazí se seznam zpráv ve frontě s informacemi o nich jako je hodnota putdate, puttime, a formát zprávy.

Maximální počet zpráv uložených v seznamu je 99. Je-li ve frontě více zpráv, než je tato, zobrazí se také aktuální hloubka fronty. Chcete-li vybrat zprávu pro zobrazení, zadejte číslo zprávy do vstupního pole (výchozí hodnota je 01). Pokud vaše položka není platná, obdržíte odpovídající chybovou zprávu.

Program pro obsah zprávy

Zobrazí se obsah zprávy.

Obsah je formátován a rozdělen do dvou částí:

1. deskriptor zprávy
2. Vyrovnávací paměť zprávy

Deskriptor zpráv zobrazuje obsah každého pole na samostatném řádku.

Vyrovnávací paměť zpráv je formátována v závislosti na jejím obsahu. Pokud vyrovnávací paměť obsahuje záhlaví nedoručené zprávy (MQDLH) nebo záhlaví přenosové fronty (MQXQH), jsou tyto formátovány a zobrazeny před vlastní vyrovnávací pamětí.

Před formátováním dat vyrovnávací paměti se v řádku s titulkem zobrazí délka vyrovnávací paměti zprávy v bajtech. Maximální velikost vyrovnávací paměti je 32768 bajtů a jakákoli zpráva delší, než je tato, je oříznuta. Zobrazí se úplná velikost vyrovnávací paměti spolu se zprávou označující, že se zobrazí pouze prvních 32768 bajtů zprávy.

Data vyrovnávací paměti jsou formátována dvěma způsoby:

1. Po vytištění offsetu do vyrovnávací paměti se data vyrovnávací paměti zobrazí hexadecimálně.
2. Data vyrovnávací paměti se poté zobrazí znovu jako hodnoty EBCDIC. Pokud nějakou hodnotu EBCDIC nelze vytisknout, vytiskne místo ní tečku (.).

Můžete zadat D pro odstranění, nebo F pro postoupení do pole akce. Rozhodnete-li se předat zprávu, musí být správně nastavena hodnota `forward-to queue` a `queue manager name` . Předvolby pro tato pole se čtou z polí fronty zpráv ReplyToQ a ReplyToQMgr.

Pokud přepošlete zprávu, bude blok záhlaví uložený ve vyrovnávací paměti oddělen. Je-li zpráva úspěšně předána, je odebrána z původní fronty. Zadáte-li neplatné akce, zobrazí se chybové zprávy.

K dispozici je také ukázkový panel nápovědy s názvem CSQ4CHP9 .

z/OS *Ukázka Asynchronous Put na systému z/OS*

Ukázkový program Asynchronous Put vkládá zprávy do fronty s použitím asynchronního volání MQPUT. Ukázka také načítá informace o stavu pomocí volání MQSTAT.

Aplikace Asynchronous Put používají tato volání MQI:

- MQCONN
- MQOPEN
- MQPUT
- MQSTAT
- MQCLOSE
- MQDISC

Vzorové programy jsou dodávány v programovacím jazyce C.

Aplikace Asynchronní vkládání se spouštějí v prostředí dávkového zpracování. Dávkové aplikace viz Další ukázky .

Toto téma také poskytuje informace o návrhu programu Asynchronní spotřeba a spuštění ukázky CSQ4BCS2 .

- “Spuštění ukázky CSQ4BCS2” na stránce 1187
- “Návrh ukázkového programu Asynchronous Put” na stránce 1187

Spuštění ukázky CSQ4BCS2

Tento ukázkový program má až šest parametrů:

1. Název cílové fronty (povinné).
2. Název správce front (volitelné).
3. Volby otevření (volitelné).
4. Volby zavření (volitelné).
5. Název cílového správce front (volitelné).
6. Název dynamické fronty (volitelné).

Není-li správce front zadán, připojí se CSQ4BCS2 k výchozímu správci front. Obsah zprávy se poskytuje prostřednictvím standardního vstupu (**SYSIN DD**).

K dispozici je ukázkový JCL ke spuštění programu, který se nachází v umístění CSQ4BCSP.

Návrh ukázkového programu Asynchronous Put

Program používá volání MQOPEN buď s dodanými volbami výstupu, nebo s volbami MQOO_OUTPUT a MQOO_FAIL_IF_QUIESCING, aby bylo možné otevřít cílovou frontu pro vkládání zpráv.

Pokud program nemůže otevřít frontu, výstupem programu je chybová zpráva obsahující kód příčiny vrácený voláním MQOPEN. Chcete-li program ponechat na tomto a dalším volání MQI jednoduché, jsou pro mnohé z voleb použity výchozí hodnoty.

Pro každý řádek vstupu program přečte text do vyrovnávací paměti a použije volání MQPUT s MQPMO_ASYNC_RESPONSE k vytvoření zprávy datagramu obsahující text této řádky a asynchronně vloží zprávu do cílové fronty. Program pokračuje, dokud nedosáhne konce vstupu, nebo dokud se volání MQPUT nezdaří. Pokud se program dostane na konec vstupu, zavře frontu pomocí volání MQCLOSE.

Program pak vydá volání MQSTAT, která vrátí strukturu MQSTS, a zobrazí zprávy obsahující počet úspěšně vložených zpráv, počet zpráv vložených s varováním a počet selhání.

Poznámka: Chcete-li zjistit, co se stane, když MQSTAT zjistí chybu MQPUT, nastavte parametr MAXDEPTH na cílovou frontu na nízkou hodnotu.

z/OS *The Batch Asynchronous Consumption sample on z/OS*

Ukázkový program CSQ4BCS1 je dodáván v jazyce C, který umožňuje asynchronní příjem zpráv z více front pomocí funkce MQCB a MQCTL.

Ukázky Asynchronous Consumption se spouštějí v dávkovém prostředí. Dávkové aplikace viz Další ukázky .

Existuje také ukázka COBOL, která se spouští v prostředí produktu CICS , viz “Ukázka CICS Asynchronní spotřeba a publikování/odběr v systému z/OS” na stránce 1189.

Aplikace používají tato volání MQI:

- MQCONN
- MQOPEN
- MQCLOSE
- MQDISC

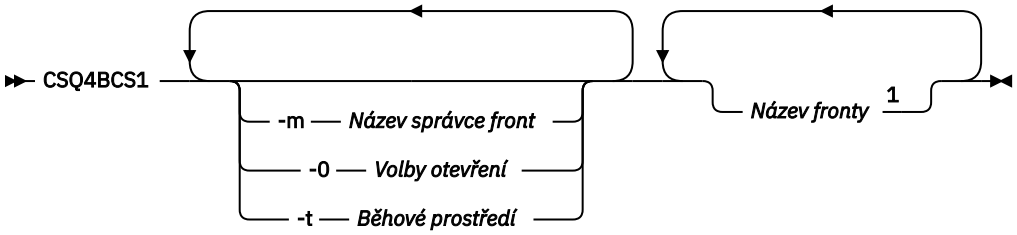
- MQCB
- MQCTL

Toto téma také poskytlo informace o následujících záhlavích:

- “Spuštění ukázky CSQ4BCS1” na stránce 1188
- “Návrh ukázkového programu Dávková asynchronní spotřeba” na stránce 1188

Spuštění ukázky CSQ4BCS1

Tento ukázkový program se řídí touto syntaxí:



Poznámky:

[1] Tato ukázka podporuje maximálně 10 front. Chcete-li číst zprávy z více front, zadejte více než jeden název fronty.

Pro spuštění tohoto programu je k dispozici ukázka JCL, je umístěn v souboru CSQ4BCSC.

Návrh ukázkového programu Dávková asynchronní spotřeba

Ukázka ukazuje, jak číst zprávy z více front v pořadí jejich příchodu. To bude vyžadovat více kódu pomocí synchronního příkazu MQGET. Při asynchronní spotřebě není požadován žádný systém výzev a správa podprocesů a úložišť je prováděna produktem IBM MQ. V ukázkovém programu jsou chyby zapsány do konzoly.

Ukázkový kód má následující kroky:

1. Definujte jednotlivou funkci zpětného volání spotřeby zpráv.

```
void MessageConsumer(MQHCONN   hConn,
MQMD  * pMsgDesc,
MQGMO  * pGetMsgOpts,
MQBYTE * Buffer,
MQCBC  * pContext)
{ ... }
```

2. Připojte se ke správci front.

```
MQCONN(QMName,&Hcon,&CompCode,&CReason);
```

3. Otevřete vstupní fronty a přidružte každou frontu k funkci zpětného volání `MessageConsumer` .

```
MQOPEN(Hcon,&od,O_options,&Hobj,&OpenCode,&Reason);
cbd.CallbackFunction = MessageConsumer;
MQCB(Hcon,MQOP_REGISTER,&cbd,Hobj,&md,&gmo,&CompCode,&Reason);
```

`cbd.CallbackFunction` není třeba nastavovat pro každou frontu; jedná se pouze o vstupní pole. Ke každé frontě můžete přidružit jinou funkci zpětného volání.

4. Spustit spotřebu zpráv.

```
MQCTL(Hcon,MQOP_START,&ctlo,&CompCode,&Reason);
```

5. Počkejte, až uživatel stiskne klávesu Enter, a pak zastavte spotřebu zpráv.

```
MQCTL(Hcon,MQOP_STOP,&ctlo,&CompCode,&Reason);
```

6. Nakonec se odpojte od správce front.

```
MQDISC(&Hcon,&CompCode,&Reason);
```

z/OS *Ukázka CICS Asynchronní spotřeba a publikování/odběr v systému z/OS*

Ukázkové programy Asynchronous Consumption a Publish/Subscribe demonstrují použití asynchronní spotřeby a funkce publikování a odběru v rámci produktu CICS.

Program *Registration client* registruje tři obslužné rutiny zpětného volání (obslužná rutina událostí a dva spotřebitele zpráv) a spouští asynchronní spotřebu. Program *Klient systému zpráv* vkládá zprávy do fronty nebo publikuje vhodné zprávy z konzoly produktu CICS ke spotřebě dvěma spotřebiteli zpráv (CSQ4CVCN a CSQ4CVCT).

Chcete-li zajistit běhovou kontrolu nad chováním ukázky, lze jednomu ze spotřebitelů zpráv získat pokyny prostřednictvím zpráv, které obdrží, se SUSPENEND, RESUME nebo DEREGISTER kterýchkoli z obslužných rutin Callback. Lze jej také použít k vydání příkazu MQCTL STOP k ukončení asynchronní spotřeby pod kontrolou. Další spotřebitel zpráv je registrován pro přihlášení k odběru tématu.

Každý program vydá v příslušných bodech příkazy COBOL DISPLAY, aby se zobrazily chování ukázky.

Aplikace používají tato volání MQI:

- MQOPEN
- MQPUT
- MQSUB
- MQGET
- MQCLOSE
- MQCB
- MQCTL

Programy jsou dodávány v jazyce COBOL. Viz ukázky CICS Asynchronous Consumption and Publish/Subscribe samples pro aplikace produktu CICS .

Toto téma také poskytuje následující informace:

- “Nastavení” na stránce 1189
- “Registrační klient CSQ4CVRG” na stránce 1190
- “Obslužná rutina událostí CSQ4CVEV” na stránce 1190
- “Zákazník jednoduché zprávy CSQ4CVCN” na stránce 1190
- “Kontrola spotřebitele zpráv CSQ4CVCT” na stránce 1190
- “Klient systému zpráv CSQ4CVPT” na stránce 1190

Nastavení

Názvy fronty a tématu používané Konzumenty zpráv jsou pevně naprogramovány v programech Registrace a v programech klienta systému zpráv.

Frontu, **SAMPLE.CONTROL.QUEUE**, měla by být definována ve správci front přidruženému k oblasti CICS před spuštěním ukázky. Téma **Novinky/média/Filmy**lze definovat, je-li to vyžadováno, nebo je vytvořeno za běhu pod výchozím administrativním objektem, pokud tento objekt neexistuje.

Programy CICS a definice transakcí lze instalovat instalováním skupiny: CSQ4SAMP.

Registrační klient CSQ4CVRG

Program Registration Client musí být spuštěn v rámci transakce CICS v rámci transakce MVRG. Neznamená to žádný vstup.

Při spuštění registruje klient registrace následující obslužné rutiny zpětného volání pomocí funkce MQCB:

- CSQ4CVEV jako Obslužná rutina událostí.
- CSQ4CVCN jako konzument zpráv v tématu **Novinky/média/Filmy**.
- CSQ4CVCT jako Spotřebitel zpráv ve frontě **SAMPLE.CONTROL.QUEUE**.

Registrační klient předává datovou strukturu obsahující názvy všech tří registrovaných obslužných rutin Callback do CSQ4CVCTspolu s manipulátory objektů přidruženými ke dvěma spotřebitelům zpráv.

Po registraci obslužných rutin Callback vydá klient registrace k zahájení asynchronní spotřeby MQCTL START_WAIT a pozastaví řízení, dokud se do řízení nevrátí řízení (například jedním ze obslužných rutin Callback, které vydávají příkaz MQCTL STOP).

Obslužná rutina událostí CSQ4CVEV

Když je řízen, obslužná rutina událostí zobrazí zprávu označující typ volání (např. START). Při použití kódu příčiny IBM MQ s kódem příčiny CONNECTION_QUIESCING vydává obslužná rutina událostí MQCTL STOP pro ukončení asynchronní spotřeby a návrat řízení do registračního klienta.

Zákazník jednoduché zprávy CSQ4CVCN

Je-li tato zpráva zobrazena, zobrazí se zpráva informující o typu volání (například REGISTER). Při použití typu volání MSG_REMOVED načte Konzument zpráv příchozí zprávu a předává ji do protokolu úlohy produktu CICS .

Kontrola spotřebitele zpráv CSQ4CVCT

Je-li tato zpráva zobrazena, zobrazí se zpráva informující o typu volání (například START). Při použití typu volání MSG_REMOVED načte Konzument Message zprávu příchozí zprávu a datovou strukturu předanou registračním klientem. Na základě obsahu zprávy vydává příslušné příkazy MQCB nebo MQCTL na jednu z následujících možností:

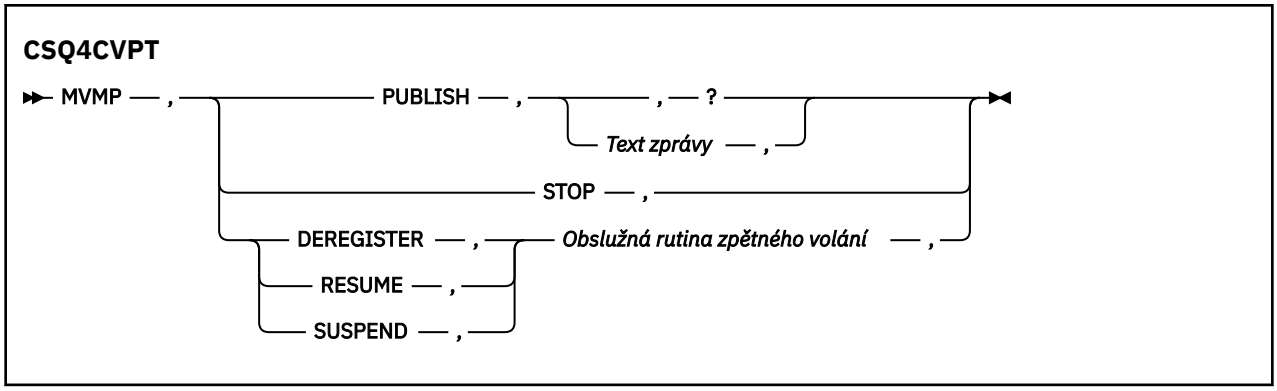
- ZASTAVIT asynchronní spotřebu (vrací řízení do registračního klienta).
- SUSPEND, RESUME nebo DEREGISTER a pojmenované obslužné rutiny zpětného volání (včetně samotného).

Klient systému zpráv CSQ4CVPT

Klient systému zpráv má dvě funkce:

- Publikuje zprávu do tématu ke spotřebě Konzumentem zpráv CSQ4CVCN.
- Zavede řídicí zprávu do fronty ke spotřebě produktem Control Message Consumer CSQ4CVCT, což má za následek potenciální změnu v chování ukázky.

Program klienta systému zpráv musí být spuštěn z konzoly produktu CICS v rámci transakce produktu CICS a má vstup příkazového řádku s touto syntaxí:



PUBLISH
Publikujte text zprávy (nebo výchozí zprávu) jako Zachovaná zpráva pro spotřebu Jednoduchým konzumentem zpráv.

ZASTAVIT
Zastavit asynchronní spotřebu.

ZRUŠIT REGISTRACI
Zrušte registraci pojmenované obslužné rutiny zpětného volání.

RESUME
Obnovte pojmenovanou obslužnou rutinu zpětného volání.

SUSPEND
Pozastavit uvedenou obslužnou rutinu zpětného volání.

Vstupní pole jsou poziční a jsou oddělena čárkami. Klíčová slova a názvy obslužných rutin zpětného volání nerozlišují velikost písmen.

Příklady:

Tabulka 175. Příklady vstupů

Příklad	**Popis**
MVMP, PUBLISH,,	Publikovat výchozí zprávu
MVMP,publish, `A short message`,	Publikovat daný text
MVMP, STOP,	Zastavit asynchronní spotřebu
MVMP,DEREGISTER,CSQ4CVEV,	Zrušit registraci obslužné rutiny událostí
MVMP,resume,csq4cvcn,	Obnovit spotřebitele jednoduchých zpráv
MVMP,SUSPEND,CSQ4CVEV,	Pozastavit obslužnou rutinu událostí

Kde MVMP je transakce CICS přidružená k programu klienta systému zpráv CSQ4CVPT.

Poznámka:

- Pozastavení nebo zrušení registrace všech obslužných rutin Callback končí START_WAIT vydaným klientem registrace, vrací k němu obslužný prvek a ukončuje úlohu.
- Pozastavení nebo zrušení registrace obslužné rutiny pro řízení zpětného volání se záměrně nezabránilo, ale odebírá schopnost dále kontrolovat chování ukázky.

z/OS *Ukázka Publikování/odběr v systému z/OS*

Ukázkové programy publikování/odběru demonstrují použití funkcí publikování a odběru v produktu IBM MQ.

K dispozici jsou čtyři programy v jazyku COBOL a dva ukázkové programy programovacího jazyka COBOL, které demonstrují, jak programovat na rozhraní Publikování/odběru produktu IBM MQ . Programy jsou

dodávány v jazyce C a COBOL. Aplikace se spouštějí v dávkovém prostředí; viz téma Ukázky publikování/odběru pro dávkové aplikace.

Existují také ukázky jazyka COBOL, které se spouštějí v prostředí produktu CICS , viz “Ukázka CICS Asynchronní spotřeba a publikování/odběr v systému z/OS” na stránce 1189.

Toto téma také poskytuje informace o tom, jak spustit ukázkové programy publikování/odběru. Tyto ukázkové programy zahrnují:

Spuštění ukázky CSQ4BCP1

Tento program je napsán v C; publikuje zprávy do tématu. Před spuštěním tohoto programu spusťte jeden z ukázek odběratele.

Tento program má až čtyři parametry:

1. Název cílového řetězce tématu (povinné).
2. Název správce front (volitelné).
3. Volby otevření (volitelné).
4. Volby zavření (volitelné).

Není-li správce front zadán, připojí se CSQ4BCP1 k výchozímu správci front. K dispozici je ukázkový JCL ke spuštění programu, který je umístěn v CSQ4BCPP.

Obsah zprávy se poskytuje prostřednictvím standardního vstupu (**SYSIN DD**).

Spuštění ukázky CSQ4BCP2

Tento program je napsán v C; přihlásí se k odběru tématu a vytiskne přijaté zprávy.

Tento program bude mít až tři parametry:

1. Název cílového řetězce tématu (povinné).
2. Název správce front (volitelné).
3. Volby odběru MQSD (volitelné).

Není-li správce front zadán, připojí se CSQ4BCP2 k výchozímu správci front. K dispozici je ukázka JCL pro spuštění programu, který je umístěn v souboru CSQ4BCPS.

Spuštění ukázky CSQ4BCP3

Tento program je napsán v C; přihlašuje se k odběru tématu pomocí uživatelsky zadané cílové fronty a tiskne přijaté zprávy.

Tento program má až čtyři parametry:

1. Název cílového řetězce tématu (povinné).
2. Název místa určení (povinné).
3. Název správce front (volitelné).
4. Volby odběru MQSD (volitelné).

Není-li správce front zadán, připojí se CSQ4BCP3 k výchozímu správci front. Pro spuštění programu existuje ukázka JCL, je umístěn v souboru CSQ4BCPD.

Spuštění ukázky CSQ4BCP4

Tento program je napsán v jazyce C; odebírá a získává zprávy z tématu, které umožňuje použití rozšířených voleb na volání MQSUB, rozšíření dostupných možností na zjednodušeném vzorku MQSUB: CSQ4BCP2. Kromě informačního obsahu zprávy jsou přijaty a zobrazeny vlastnosti zprávy pro každou zprávu.

Tento program má proměnnou sadu parametrů:

- **-t** *Topic string* (povinné).
- **-o** *Topic object name* (povinné).
- **-m** *Queue manager name* (volitelné).
- **-q** *Destination queue name* (volitelné).
- **-w** *Wait interval on MQGET in seconds* (volitelné), kde *sekundy* mohou mít některou z následujících hodnot:
  - `unlimited`: MQWI_UNLIMITED.
  - none: Bez čekání
  - *n*: Interval čekání v sekundách
  - Nebyla zadána žádná hodnota: Není-li zadána žádná hodnota, je výchozí hodnota 30 sekund.
- **-d** *Subscription name* (volitelné). Vytvoří nebo obnoví pojmenovaný trvalý odběr.
- **-k** (volitelné). Udržuje trvalý odběr v MQCLOSE.

Není-li správce front zadán, připojí se k výchozímu správci front CSQ4BCP4 . Pro spuštění programu existuje ukázka JCL, je umístěn v souboru CSQ4BCPE.

Spuštění ukázky CSQ4BVP1

Tento program je napsán v jazyce COBOL, publikuje zprávy do tématu. Před spuštěním tohoto programu spusťte jeden z ukázek odběratele.

Tento program nemá žádné parametry. Produkt **SYSIN DD** poskytuje název vstupního tématu, název správce front a obsah zprávy.

Není-li správce front zadán, připojí se CSQ4BVP1 k výchozímu správci front. K dispozici je ukázkový JCL ke spuštění programu, který je umístěn v CSQ4BVPP.

Spuštění ukázky CSQ4BVP2

Tento program je napsán v jazyce COBOL, přihlásil se k odběru tématu a vytiskne přijaté zprávy.

Tento program nemá žádné parametry. Produkt **SYSIN DD** poskytuje vstup pro název tématu a název správce front.

Není-li správce front zadán, připojí se CSQ4BVP1 k výchozímu správci front. K dispozici je ukázkový JCL ke spuštění programu, který je umístěn v CSQ4BVPP.

z/OS *Ukázka vlastnosti Message Set a Inquire na objektu z/OS*

Ukázkové programy vlastností zprávy demonstrují přidání uživatelsky definovaných vlastností do popisovače zpráv a inkvizici vlastností přidružených k této zprávě.

Aplikace používají tato volání MQI:

- MQCONN
- MQOPEN
- MQPUT
- MQGET
- MQCLOSE
- MQDISC

- MQCRTM
- MQDLTMH
- MQINQMP
- MQSETMP

Programy jsou dodávány v jazyce C. Aplikace se spouštějí v dávkovém prostředí. Dávkové aplikace viz Další ukázky .

Program CSQ4BCM1 se používá k dotazům na vlastnosti popisovače zpráv z fronty zpráv a je příkladem použití volání MQINQMP API. Ukázka získá jednu zprávu z fronty a poté vytiskne všechny vlastnosti popisovače zpráv.

Program CSQ4BCM2 se používá k nastavení vlastností obsluhy zpráv ve frontě zpráv a je příkladem použití volání rozhraní API MQSETMP. Ukázka vytvoří popisovač zprávy a vloží jej do pole `MsgHandle` struktury MQGMO. Pak vloží zprávu do fronty.

Další příklady dotazovacích a tiskových vlastností zpráv jsou zahrnuty v ukázkových programech CSQ4BCG1 a CSQ4BCP4 .

Toto téma také poskytuje informace o spuštění ukázek vlastností sady a zjišťování vlastností zpráv v následujících záhlavích:

- “Spuštění ukázky CSQ4BCM1” na stránce 1194
- “Spuštění ukázky CSQ4BCM2” na stránce 1194

Spuštění ukázky CSQ4BCM1

Tento program má až čtyři parametry:

1. Název cílové fronty (povinné).
2. Název správce front (volitelné).
3. Volby otevření (volitelné).
4. Volby zavření (volitelné).

Spuštění ukázky CSQ4BCM2

Tento program má až šest parametrů:

1. Název cílové fronty (povinné).
2. Název správce front (volitelné).
3. Volby otevření (volitelné).
4. Volby zavření (volitelné).
5. Název cílového správce front (volitelné).
6. Název dynamické fronty (volitelné).

Názvy vlastností, hodnoty a obsah zprávy jsou poskytovány prostřednictvím standardního vstupu (**SYSIN DD**). K dispozici je ukázkový soubor JCL pro spuštění programu, který je umístěn v umístění CSQ4BCMP.

Vyvíjení aplikací pro MQ Telemetry

Telemetrické aplikace integrují smyslová a řídicí zařízení s dalšími zdroji informací, které jsou k dispozici na internetu a v podnicích.

Vyvíjejte aplikace pro produkt MQ Telemetry s využitím vzorů návrhu, příkladů použití, ukázkových programů, koncepcí programování a referenčních informací.

Související informace

MQ Telemetry

Příklady použití Telemetrie

instalaceMQ Telemetry
Správa serveruMQ Telemetry
Referenční informace k produktu MQ Telemetry
MQ Telemetry odstraňování problémů

IBM MQ Telemetry Transport ukázkové programy

K dispozici jsou ukázkové skripty, které pracují s ukázkovou aplikací klienta IBM MQ Telemetry Transport v3 (`mqttv3app.jar`). Pro databázi IBM MQ 8.0.0 a novější již ukázková aplikace klienta není zahrnuta do produktu MQ Telemetry. Bylo součástí (již není k dispozici) IBM Messaging Telemetry Clients SupportPac. Podobné ukázkové aplikace jsou i nadále volně dostupné z prostředí Eclipse Paho a MQTT.org.

Nejnovější informace a soubory ke stažení najdete v následujících zdrojích:

- Projekt Eclipse Paho a MQTT.org mají k dispozici bezplatné stažení nejnovějších klientů telemetrie a ukázek pro řadu programovacích jazyků. Pomocí těchto stránek můžete vyvíjet ukázkové programy pro publikování a odběr produktu IBM MQ Telemetry Transport a pro přidávání funkcí zabezpečení.
- IBM Messaging Telemetry Clients SupportPac již není k dispozici ke stažení. Máte-li dříve staženou kopii, má následující obsah:
  - Verze MA9B IBM Messaging Telemetry Clients SupportPac zahrnovala kompilovanou ukázkovou aplikaci (`mqttv3app.jar`) a přidruženou knihovnu klienta (`mqttv3.jar`). Byly poskytnuty v těchto adresářích:
    - `ma9b/SDK/clients/java/org.eclipse.paho.sample.mqttv3app.jar`
    - `ma9b/SDK/clients/java/org.eclipse.paho.client.mqttv3.jar`
  - Ve verzi MA9C tohoto balíku SupportPac byl odebrán adresář `/SDK/` a obsah:
    - Byl poskytnut pouze zdroj pro ukázkovou aplikaci (`mqttv3app.jar`). Bylo to v tomto adresáři:

      ```
      ma9c/clients/java/samples/org/eclipse/paho/sample/mqttv3app/*.java
      ```

    - Byla stále poskytnuta kompilovaná knihovna klienta. Bylo to v tomto adresáři:

      ```
      ma9c/clients/java/org.eclipse.paho.client.mqttv3-1.0.2.jar
      ```

Pokud stále máte kopii (již není k dispozici) IBM Messaging Telemetry Clients SupportPac, informace o instalaci a spuštění ukázkové aplikace jsou k dispozici v tématu Ověření instalace produktu MQ Telemetry pomocí příkazového řádku.

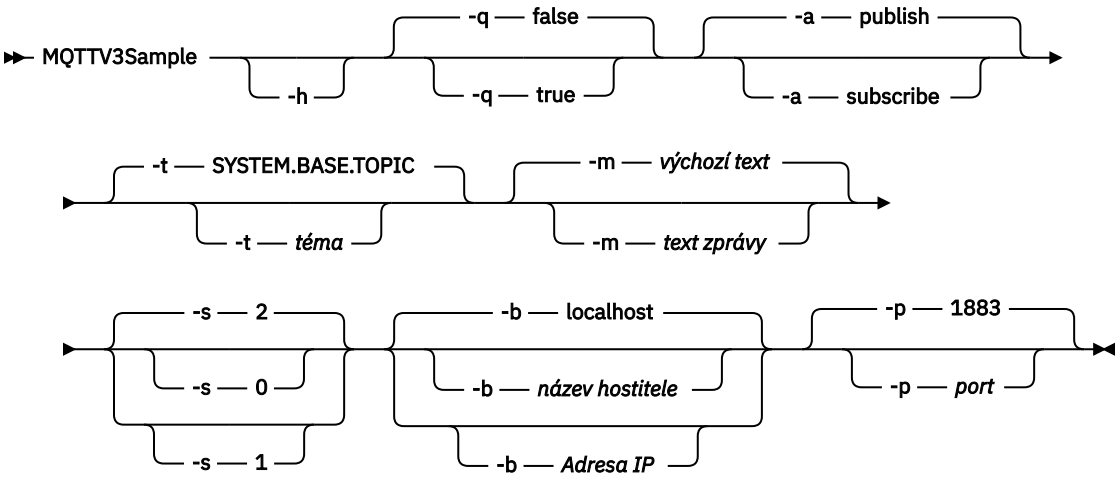
Program MQTTV3Sample

Referenční informace o ukázkové syntaxi a parametrech pro program MQTTV3Sample .

Účel

Program MQTTV3Sample lze použít k publikování zprávy a k přihlášení k odběru tématu.

Syntaxe MQTTV3Sample



Parametry

-h
Vytisknout tento text nápovědy a ukončit

-q
Nastavte tichý režim místo použití výchozího režimu false.

-a
Nastavte publikování nebo odběr, místo toho, že byste předpokládali výchozí akci publikování.

-t
Publikování nebo odběr tématu, místo publikování nebo odběru výchozího tématu

-m
Publikujte text zprávy místo odeslání výchozího textu publikace, "Ahoj z aplikace MQTT v3 ".

-s
Nastavte QoS místo použití předvolené QoS, 2.

-b
Připojte se k tomuto názvu hostitele nebo k adrese IP namísto připojení k výchozímu názvu hostitele localhost.

-p
Použijte tento port místo použití výchozího nastavení, 1883.

Spusťte program MQTTV3Sample

Chcete-li se přihlásit k odběru tématu v systému Windows, použijte následující příkaz:

```
runMQTTV3Sample -a subscribe
```

Chcete-li publikovat zprávu na serveru Windows, použijte příkaz:

```
runMQTTV3Sample
```

Další informace o spuštění poskytnutých ukázkových skriptů viz “IBM MQ Telemetry Transport ukázkové programy” na stránce 1195.

Koncepce programování klienta MQTT

Koncepty popsané v tomto oddílu vám pomohou pochopit knihovny klienta pro MQTT protocol. Koncepty doplňují dokumentaci k rozhraní API doprovázející knihovny klienta.

Nejnovější informace a soubory ke stažení najdete v následujících zdrojích:

- Projekt Eclipse Paho a MQTT.org mají k dispozici bezplatné stažení nejnovějších klientů telemetrie a ukázek pro řadu programovacích jazyků. Pomocí těchto stránek můžete vyvíjet ukázkové programy pro publikování a odběr produktu IBM MQ Telemetry Transport a pro přidávání funkcí zabezpečení.
- IBM Messaging Telemetry Clients SupportPac již není k dispozici ke stažení. Máte-li dříve staženou kopii, má následující obsah:
  - Verze MA9B IBM Messaging Telemetry Clients SupportPac zahrnovala kompilovanou ukázkovou aplikaci (`mqttv3app.jar`) a přidruženou knihovnu klienta (`mqttv3.jar`). Byly poskytnuty v těchto adresářích:
    - `ma9b/SDK/clients/java/org.eclipse.paho.sample.mqttv3app.jar`
    - `ma9b/SDK/clients/java/org.eclipse.paho.client.mqttv3.jar`
  - Ve verzi MA9C tohoto balíku SupportPac byl odebrán adresář `/SDK/` a obsah:
    - Byl poskytnut pouze zdroj pro ukázkovou aplikaci (`mqttv3app.jar`). Bylo to v tomto adresáři:

      ```
      ma9c/clients/java/samples/org/eclipse/paho/sample/mqttv3app/*.java
      ```

    - Byla stále poskytnuta kompilovaná knihovna klienta. Bylo to v tomto adresáři:

      ```
      ma9c/clients/java/org.eclipse.paho.client.mqttv3-1.0.2.jar
      ```

Chcete-li vyvinout a spustit klienta MQTT , musíte tyto prostředky zkopírovat nebo instalovat na klientské zařízení. Není třeba instalovat oddělené běhové prostředí klienta.

Podmínky licencování pro klienty jsou přidruženy k serveru, ke kterému připojujete klienty.

Klientské knihovny produktu MQTT jsou referenční implementace produktu MQTT protocol. Můžete implementovat vlastní klienty v různých jazycích vhodných pro různé platformy zařízení. Viz IBM MQ Telemetry Transport format and protocol.

Dokumentace k rozhraním API nečiní žádné předpoklady o tom, ke kterému serveru MQTT je klient připojen. Chování klienta se může mírně lišit, pokud je připojeno k různým serverům. Následující popisy popisují chování klienta při připojení ke službě telemetrie IBM MQ .

Zpětná volání a synchronizace v klientských aplikacích produktu MQTT

Programovací model klienta produktu MQTT rozsáhle používá podprocesy. Podprocesy oddělující aplikaci klienta MQTT , stejně jako mohou, z prodlev při přenosu zpráv na server a ze serveru. Publikování, tokeny doručení a ztracené události připojení se doručují do metod ve třídě zpětného volání, která implementuje `MqttCallback`.

Zpětná volání

Poznámka: Nejnovější změny produktu `MqttCallback`najdete na webových stránkách Eclipse Paho . Například `MqttCallback` je definován jako Rozhraní v verzi Paho klienta a asynchronní metody jsou poskytovány třídou Paho `MqttAsyncClient` .

Rozhraní `MqttCallback` má tři metody zpětného volání:

`connectionLost(java.lang.Throwable cause)`

`connectionLost` je volán, když komunikační chyba vede k uvolnění spojení. Nazývá se také tehdy, když server po navázání spojení přeruší spojení v důsledku chyby na serveru. Chyby serveru se protokolují do protokolu chyb správce front. Server zruší spojení s klientem a klient zavolá `MqttCallback.connectionLost`.

Jediné vzdálené chyby, vyvolané jako výjimky ze stejného podprocesu jako klientská aplikace, jsou výjimky z produktu `MqttClient.connect`. Chyby zjištěné serverem po zavedení připojení jsou nahlášeny zpět do metody zpětného volání `MqttCallback.connectionLost` jako `throwables`.

Typické chyby serveru, které vedou k chybě `connectionLost` , jsou chyby autorizace. Server telemetrie se například pokusí publikovat na téma jménem klienta, který není autorizován

k publikování v rámci daného tématu. Vše, co má za následek vrácení kódu podmínky produktu MQCC_FAIL do serveru telemetrie, může vést k tomu, že připojení bude zrušeno.

deliveryComplete(IMqttDeliveryToken token)

deliveryComplete je volán klientem MQTT k předání tokenu doručení klientské aplikaci, viz “Tokeny doručení” na stránce 1203. Při použití tokenu doručení může zpětné volání přistupovat k publikované zprávě s použitím metody token.getMessage.

Když zpětné volání aplikace vrátí řízení klientovi MQTT po volání metody deliveryComplete , doručení je dokončeno. Dokud nebude dokončeno doručení, zprávy s produktem QoS 1 nebo 2 jsou zachovány podle třídy perzistence.

Volání do produktu deliveryComplete je bodem synchronizace mezi aplikací a třídou perzistence. Metoda deliveryComplete se nikdy nevolá dvakrát pro stejnou zprávu.

Když se zpětné volání aplikace vrátí z produktu deliveryComplete na klienta MQTT , klient zavolá příkaz MqttClientPersistence.remove pro zprávy s hodnotou QoS 1 nebo 2. Příkaz MqttClientPersistence.remove odstraní lokálně uloženou kopii publikované zprávy.

Z pohledu zpracování transakce je volání na produkt deliveryComplete jednofázovou transakcí, která potvrzuje doručení. Pokud zpracování selže během zpětného volání, při restartu klienta MqttClientPersistence.remove se znovu volá a odstraní lokální kopie publikované zprávy. Zpětné volání se nevolá znovu. Pokud používáte zpětné volání k uložení protokolu doručených zpráv, nemůžete synchronizovat protokol s klientem produktu MQTT . Chcete-li uložit protokol spolehlivě, aktualizujte protokol ve třídě MqttClientPersistence .

Na token doručení a na zprávu odkazuje hlavní aplikační podproces a klient produktu MQTT . Klient MQTT zruší odkaz na objekt MqttMessage při dokončení doručení a objekt tokenu doručení, když se klient odpojí. Objekt MqttMessage může být vyčištěn odpad po dokončení doručení, pokud aplikace klienta zruší odkaz na něj. Po odpojení relace může být token doručení odebrán.

Po publikování zprávy můžete získat atributy IMqttDeliveryToken a MqttMessage po publikování zprávy. Pokusíte-li se nastavit libovolný atribut MqttMessage poté, co byla zpráva publikována, výsledek není definován.

Klient MQTT pokračuje v zpracování potvrzení doručení, pokud se klient znovu připojí k předchozí relaci se stejnou hodnotou ClientIdentifier ; viz “Vyčistit relace” na stránce 1200. Aplikace klienta MQTT musí nastavit MqttClient.CleanSession na false pro předchozí relaci a nastavit ji na false v nové relaci. Klient produktu MQTT vytváří nové tokeny doručení a objekty zpráv v nové relaci pro nevyřízené doručení. Zotavuje objekty pomocí třídy MqttClientPersistence . Pokud má klient aplikace stále odkazy na staré tokeny doručení a zprávy, vyhodnoťte je. Zpětné volání aplikace se volá v nové relaci pro všechny doručení zahájené v předchozí relaci a dokončené v této relaci.

Zpětné volání aplikace se volá po připojení klienta aplikace, když je dokončeno nevyřízené doručení. Než se klient aplikace připojí, může nevyřízené doručení načíst pomocí metody MqttClient.getPendingDeliveryTokens .

Všimněte si, že aplikace klienta původně vytvořila objekt zprávy, který je publikován, a jeho bajtové pole informačního obsahu. Klient MQTT odkazuje na tyto objekty. Objekt zprávy vrácený tokenem doručení v metodě token.getMessage nemusí nutně znamenat stejný objekt zprávy vytvořený klientem. If a new MQTT client instance re-creates the delivery token, the MqttClientPersistence class re-creates the MqttMessage object. Pro konzistenci token.getMessage vrátí hodnotu null , pokud token.isCompleted je true, bez ohledu na to, zda byl objekt zprávy vytvořen klientem aplikace nebo třídou MqttClientPersistence .

messageArrived(String topic, MqttMessage message)

messageArrived je volán při doručení publikace pro klienta, který odpovídá tématu odběru. topic je téma publikace, nikoli filtr odběru. Mohou se lišit, pokud filtr obsahuje zástupné znaky.

Pokud se téma shoduje s více odběry vytvořenými klientem, obdrží klient více kopií publikování. Pokud klient publikuje do tématu, které se také přihlásí k odběru, obdrží kopii své vlastní publikace.

Je-li zpráva odeslána s QoS z 1 nebo 2, zpráva je uložena třídou MqttClientPersistence před voláním klienta MQTT messageArrived. Příkaz messageArrived se chová jako deliveryComplete: volá se pouze jednou pro publikování a lokální kopie této publikace je odstraněna produktem MqttClientPersistence.remove , když se příkaz messageArrived

vrátí klientovi MQTT . Klient MQTT zruší své odkazy na téma a zprávu, když se `messageArrived` vrátí klientovi MQTT . Objekty tématu a zprávy jsou shromažďovány v rámci uvolňování paměti, pokud klient aplikace neuchoval odkaz na objekty.

Zpětná volání, rozdělování na podprocesy a synchronizace aplikací klienta

Klient produktu MQTT volá metodu zpětného volání na samostatný podproces do hlavního podprocesu aplikace. Klientská aplikace nevytváří podproces pro zpětné volání, je vytvořena klientem MQTT .

Klient produktu MQTT synchronizuje metody zpětného volání. V daném okamžiku je spuštěna pouze jedna instance metody zpětného volání. Synchronizace umožňuje snadno aktualizovat objekt, který obsahuje informace o tom, které publikace byly doručeny. Jedna instance serveru `MqttCallback.deliveryComplete` se spustí v daném okamžiku, a proto je bezpečné aktualizovat záznam, aniž by došlo k další synchronizaci. Je také tomu tak, že v daném okamžiku dorazí pouze jedna publikace. Váš kód v metodě `messageArrived` může aktualizovat objekt, aniž by se synchronizoval. Pokud odkazujete na záznam nebo objekt, který se právě aktualizuje, v jiném podprocesu synchronizujte záznam nebo objekt.

Token doručení poskytuje mechanizmus synchronizace mezi hlavním podprocesem aplikace a doručením publikace. Metoda `token.waitForCompletion` čeká, dokud nebude dokončeno doručení určité publikace, nebo dokud nevyprší volitelný časový limit. Produkt `token.waitForCompletion` můžete používat následujícím způsobem ke zpracování jedné publikace současně.

Proveďte synchronizaci s metodou `MqttCallback.deliveryComplete` . Pouze když se `MqttCallback.deliveryComplete` vrátí na klienta MQTT , obnoví se `token.waitForCompletion` . Pomocí tohoto mechanizmu můžete synchronizovat spouštěný kód v produktu `MqttCallback.deliveryComplete` před spuštěním kódu v hlavním podprocesu aplikace.

Co když jste chtěli publikovat bez čekání na doručení každé publikace, ale chcete potvrdit, kdy byly všechny publikace doručeny? Pokud publikujete na jednom podprocesu, poslední publikování, které má být odesláno, je také poslední, které má být doručeno.

Synchronizace požadavků odeslaných na server

Tabulka 176 na stránce 1199 popisuje metody v klientovi MQTT Java , které odesílají požadavek na server. Pokud aplikační klient nenastaví neomezenou časový limit, klient nikdy na server nečeká nekonečně dlouhou dobu. Pokud se klient zablokuje, je to buď problém programování aplikace, nebo defekt v klientovi MQTT .

Tabulka 176. Chování synchronizace metod, které vedou k požadavkům na server

Metoda	Synchronizace	Interval časového limitu
`MqttClient.Connect`	Čeká na navázání spojení se serverem.	Výchozí hodnota je 30 sekund, nebo jak je nastaveno parametrem, pak vyvolá výjimku.
`MqttClient.Disconnect`	Čeká, až klient MQTT dokončí práci, kterou musí provést, a aby se relace TCP/IP odpojicím.	
`MqttClient.Subscribe`	Čeká na dokončení metody odběru nebo odběru UnSubscribe .	
`MqttClient.UnSubscribe`		
`MqttClient.Publish`	Po předání požadavku klientovi produktu MQTT se okamžitě vrátí k podprocesu aplikace.	Není.
`IMqttDeliveryToken.waitFor Completion`	Čeká na vrácení doručovacího tokenu.	Neomezené, nebo jako parametr.

Související pojmy

Vyčistit relace
Klient produktu MQTT a služba telemetrie (MQXR) udržují informace o stavu relace. Informace o stavu se používají k zajištění doručení "alespoň jednou" a "přesně jednou" doručení a "přesně jednou" pro příjem publikací. Stav relace také zahrnuje odběry vytvořené klientem MQTT . Můžete zvolit spuštění klienta MQTT s nebo bez udržování informací o stavu mezi relacemi. Změňte režim čisté relace nastavením `MqttConnectOptions.cleanSession` před připojením.

Identifikátor klienta
Identifikátor klienta je 23bajtový řetězec, který identifikuje klienta MQTT . Každý identifikátor musí být jedinečný, tedy platný ve stejnou dobu pouze pro jediného připojeného klienta. Identifikátor musí obsahovat pouze znaky platné v názvu správce front. V rámci těchto omezení budete moci používat libovolný identifikační řetězec. Je důležité mít proceduru pro přidělování identifikátorů klientů a prostředků ke konfiguraci klienta s vybraným identifikátorem.

Tokeny doručení

Datum poslední vůle a potvrzení
Dojde-li k neočekávanému ukončení připojení klienta MQTT , můžete produkt MQ Telemetry nakonfigurovat tak, aby odeslal publikaci "last will and testament". Předefinujte obsah publikace a téma, kterému jej chcete odeslat. "Poslední vůle a testament" je vlastnost připojení. Vytvořte jej před připojením klienta.

Perzistence zpráv v klientech produktu MQTT
Zprávy publikování jsou prováděny jako trvalé, pokud jsou odeslány s kvalitou služby "alespoň jednou", nebo "přesně jednou". Na klientovi můžete implementovat vlastní mechanismus perzistence nebo použít výchozí mechanismus perzistence, který je dodáván s klientem. Perzistence funguje v obou směrech, pro publikování odeslané na klienta nebo z klienta.

Publikace
Publikace jsou instance `MqttMessage` , které jsou přidruženy k řetězci tématu. Klienti produktu MQTT mohou vytvářet publikace k odesílání do produktu IBM MQa k odběru témat v produktu IBM MQ se zasíláte publikacemi.

Kvality služby poskytované klientem MQTT
Klient produktu MQTT poskytuje tři úrovně kvality služby pro dodávání publikací do produktu IBM MQ a klientovi MQTT : "maximálně jednou", "alespoň jednou" a "přesně jednou". Když klient MQTT odešle požadavek produktu IBM MQ k vytvoření odběru, odešle se požadavek s kvalitou služby "alespoň jednou".

Zachovaná publikování a klienti MQTT
Téma může mít jednu, a pouze jednu zachovanou publikaci. Pokud vytvoříte odběr tématu, které má zachované publikování, bude vám ihned předáno publikování.

Odběry
Vytvoření odběrů pro registraci zájmu v tématech publikování pomocí filtru témat. Klient může vytvořit více odběrů nebo odběr obsahující filtr témat, který používá zástupné znaky, a zaregistrovat tak zájem o více témat. Publikace na témata, která odpovídají filtrům, se posílají na klienta. Odběry mohou zůstat aktivní, když je klient odpojen. Publikace se odesílají klientovi, jakmile se znovu připojí.

Řetězce témat a filtry témat v klientech produktu MQTT
Řetězce témat a filtry témat se používají pro publikování a odběr. Syntaxe řetězců témat a filtrů v klientech produktu MQTT je z velké části stejné jako řetězce témat v produktu IBM MQ.

Vyčistit relace

Klient produktu MQTT a služba telemetrie (MQXR) udržují informace o stavu relace. Informace o stavu se používají k zajištění doručení "alespoň jednou" a "přesně jednou" doručení a "přesně jednou" pro příjem publikací. Stav relace také zahrnuje odběry vytvořené klientem MQTT . Můžete zvolit spuštění klienta MQTT s nebo bez udržování informací o stavu mezi relacemi. Změňte režim čisté relace nastavením `MqttConnectOptions.cleanSession` před připojením.

Připojíte-li klientskou aplikaci MQTT pomocí metody `MqttClient.connect` , klient identifikuje připojení s použitím identifikátoru klienta a adresy serveru. Server kontroluje, zda byly informace o relaci uloženy z předchozího připojení k serveru. Pokud předchozí relace stále existuje a `cleanSession=true`, pak

jsou předchozí informace o relaci na klientovi a serveru vymazány. Je-li `cleanSession=false` předchozí relace obnovena. Neexistuje-li žádná předchozí relace, bude spuštěna nová relace.

Poznámka: Administrátor produktu IBM MQ může vynutit zavření otevřené relace a odstranění všech informací o relaci. Pokud klient znovu otevře relaci s produktem `cleanSession=false`, spustí se nová relace.

Publikace

Pokud použijete výchozí `MqttConnectOptions`nebo nastavíte `MqttConnectOptions.cleanSession` na `true` před připojením klienta, všechny nevyřízené doručení publikování pro klienta se odeberou, když se klient připojí.

Nastavení čisté relace nemá žádný vliv na publikace odeslané s produktem `QoS=0`. Pro `QoS=1` a `QoS=2`může použití `cleanSession=true` vést ke ztrátě publikování.

Odběry

Pokud použijete výchozí `MqttConnectOptions`nebo nastavíte `MqttConnectOptions.cleanSession` na `true` před připojením klienta, všechny staré odběry klienta se odeberou, když se klient připojí. Všechny nové odběry, které klient během relace vytvoří, jsou při odpojení odebrány.

Pokud před připojením nastavíte `MqttConnectOptions.cleanSession` na `false` , všechny odběry vytvořené klientem budou přidány ke všem odběrům, které existovaly pro klienta dříve, než se připojí. Všechny odběry zůstávají aktivní, když se klient odpojí.

Dalším přístupem k pochopení, jak atribut `cleanSession` ovlivňuje odběry, je považovat jej za modální atribut. Ve svém výchozím režimu, `cleanSession=true`, vytváří klient odběry a přijímá publikace pouze v rámci dané relace. V alternativním režimu, `cleanSession=false`, jsou odběry trvalé. Klient se může připojit a odpojit a jeho odběry zůstanou aktivní. Když se klient znovu připojí, přijme všechny nedoručené publikace. V době připojení může upravit sadu odběrů, které jsou jeho jménem aktivní.

Před připojením musíte nastavit režim `cleanSession` ; režim trvá pro celou relaci. Chcete-li změnit jeho nastavení, je třeba klienta odpojit a znovu připojit. Pokud změníte režimy z použití `cleanSession=false` na `cleanSession=true`, všechny předchozí odběry pro klienta a všechny publikace, které nebyly přijaty, budou vyřazeny.

Související pojmy

Zpětná volání a synchronizace v klientských aplikacích produktu MQTT
Programovací model klienta produktu MQTT rozsáhle používá podprocesy. Podprocesy oddělující aplikaci klienta MQTT , stejně jako mohou, z prodlev při přenosu zpráv na server a ze serveru. Publikování, tokeny doručení a ztracené události připojení se doručují do metod ve třídě zpětného volání, která implementuje `MqttCallback`.

Identifikátor klienta
Identifikátor klienta je 23bajtový řetězec, který identifikuje klienta MQTT . Každý identifikátor musí být jedinečný, tedy platný ve stejnou dobu pouze pro jediného připojeného klienta. Identifikátor musí obsahovat pouze znaky platné v názvu správce front. V rámci těchto omezení budete moci používat libovolný identifikační řetězec. Je důležité mít proceduru pro přidělování identifikátorů klientů a prostředků ke konfiguraci klienta s vybraným identifikátorem.

Tokeny doručení

Datum poslední vůle a potvrzení
Dojde-li k neočekávanému ukončení připojení klienta MQTT , můžete produkt MQ Telemetry nakonfigurovat tak, aby odeslal publikaci "last will and testament". Předefinujte obsah publikace a téma, kterému jej chcete odeslat. "Poslední vůle a testament" je vlastnost připojení. Vytvořte jej před připojením klienta.

Perzistence zpráv v klientech produktu MQTT
Zprávy publikování jsou prováděny jako trvalé, pokud jsou odeslány s kvalitou služby "alespoň jednou", nebo "přesně jednou". Na klientovi můžete implementovat vlastní mechanismus perzistence nebo použít

výchozí mechanismus perzistence, který je dodáván s klientem. Perzistence funguje v obou směrech, pro publikování odeslané na klienta nebo z klienta.

Publikace
Publikace jsou instance `MqttMessage` , které jsou přidruženy k řetězci tématu. Klienti produktu MQTT mohou vytvářet publikace k odesílání do produktu IBM MQa k odběru témat v produktu IBM MQ se zasíláte publikacemi.

Kvality služby poskytované klientem MQTT
Klient produktu MQTT poskytuje tři úrovně kvality služby pro dodávání publikací do produktu IBM MQ a klientovi MQTT : "maximálně jednou", "alespoň jednou" a "přesně jednou". Když klient MQTT odešle požadavek produktu IBM MQ k vytvoření odběru, odešle se požadavek s kvalitou služby "alespoň jednou".

Zachovaná publikování a klienti MQTT
Téma může mít jednu, a pouze jednu zachovanou publikaci. Pokud vytvoříte odběr tématu, které má zachované publikování, bude vám ihned předáno publikování.

Odběry
Vytvoření odběrů pro registraci zájmu v tématech publikování pomocí filtru témat. Klient může vytvořit více odběrů nebo odběr obsahující filtr témat, který používá zástupné znaky, a zaregistrovat tak zájem o více témat. Publikace na témata, která odpovídají filtrům, se posílají na klienta. Odběry mohou zůstat aktivní, když je klient odpojen. Publikace se odesílají klientovi, jakmile se znovu připojí.

Řetězce témat a filtry témat v klientech produktu MQTT
Řetězce témat a filtry témat se používají pro publikování a odběr. Syntaxe řetězců témat a filtrů v klientech produktu MQTT je z velké části stejné jako řetězce témat v produktu IBM MQ.

Identifikátor klienta

Identifikátor klienta je 23bajtový řetězec, který identifikuje klienta MQTT . Každý identifikátor musí být jedinečný, tedy platný ve stejnou dobu pouze pro jediného připojeného klienta. Identifikátor musí obsahovat pouze znaky platné v názvu správce front. V rámci těchto omezení budete moci používat libovolný identifikační řetězec. Je důležité mít proceduru pro přidělování identifikátorů klientů a prostředků ke konfiguraci klienta s vybraným identifikátorem.

Identifikátor klienta se používá při administraci systému MQTT . S potencionálně stovkami tisíc klientů, kteří mají spravovat, musíte být schopni rychle identifikovat konkrétního klienta. Předpokládejme například, že zařízení má vadnou funkci a vy budete informováni o tom, že zákazník zazvoní na help desk. Zákazník musí být schopen identifikovat zařízení a vy musíte být schopni korelovat tuto identifikaci se serverem, který je obvykle připojen ke klientovi.

Když procházíte přes připojení klienta MQTT , každé připojení je označeno identifikátorem klienta. Chcete-li pomoci při rozhodování o tom, jak nejlépe mapovat tento identifikátor na zařízení a server, zeptejte se na následující otázky:

- Bylo by vhodné udržovat a používat databázi, která mapuje každé zařízení na identifikátor klienta a na server?
- Může jméno zařízení identifikovat server, ke kterému je jednotka připojena?
- Potřebujete vyhledávací tabulku, která mapuje identifikátor klienta na fyzické zařízení?
- Identifikujete identifikátor klienta určité zařízení, uživatele nebo aplikaci spuštěnou na klientovi?
- Pokud zákazník nahradí vadného zařízení novým zařízením, má nové zařízení stejný identifikátor jako staré zařízení, nebo přidělíte nový identifikátor? (Změníte-li fyzické zařízení a ponecháte stejný identifikátor, nové publikace a aktivní odběry se automaticky přenesou na nové zařízení.)

Potřebujete také systém, abyste se ujistili, že identifikátory klienta jsou jedinečné, a že musíte mít spolehlivý proces pro nastavení identifikátoru na klientovi. Je-li zařízení klienta "black-box", bez uživatelského rozhraní, můžete zařízení vyrobit s identifikátorem klienta, nebo můžete mít instalaci softwaru a konfigurační proces, který konfiguruje zařízení před aktivací.

Chcete-li uchovat identifikátor krátký a jedinečný, můžete vytvořit identifikátor klienta z 48bitové adresy MAC zařízení. Pokud velikost přenosu není kritická, můžete použít zbývajících 17 bajtů, abyste usnadnili administraci této adresy.

Související pojmy

Zpětná volání a synchronizace v klientských aplikacích produktu MQTT
Programovací model klienta produktu MQTT rozsáhle používá podprocesy. Podprocesy oddělující aplikaci klienta MQTT , stejně jako mohou, z prodlev při přenosu zpráv na server a ze serveru. Publikování, tokeny doručení a ztracené události připojení se doručují do metod ve třídě zpětného volání, která implementuje `MqttCallback`.

Vyčistit relace
Klient produktu MQTT a služba telemetrie (MQXR) udržují informace o stavu relace. Informace o stavu se používají k zajištění doručení "alespoň jednou" a "přesně jednou" doručení a "přesně jednou" pro příjem publikací. Stav relace také zahrnuje odběry vytvořené klientem MQTT . Můžete zvolit spuštění klienta MQTT s nebo bez udržování informací o stavu mezi relacemi. Změňte režim čisté relace nastavením `MqttConnectOptions.cleanSession` před připojením.

Tokeny doručení

Datum poslední vůle a potvrzení
Dojde-li k neočekávanému ukončení připojení klienta MQTT , můžete produkt MQ Telemetry nakonfigurovat tak, aby odeslal publikaci "last will and testament". Předefinujte obsah publikace a téma, kterému jej chcete odeslat. "Poslední vůle a testament" je vlastnost připojení. Vytvořte jej před připojením klienta.

Perzistence zpráv v klientech produktu MQTT
Zprávy publikování jsou prováděny jako trvalé, pokud jsou odeslány s kvalitou služby "alespoň jednou", nebo "přesně jednou". Na klientovi můžete implementovat vlastní mechanismus perzistence nebo použít výchozí mechanismus perzistence, který je dodáván s klientem. Perzistence funguje v obou směrech, pro publikování odeslané na klienta nebo z klienta.

Publikace
Publikace jsou instance `MqttMessage` , které jsou přidruženy k řetězci tématu. Klienti produktu MQTT mohou vytvářet publikace k odesílání do produktu IBM MQa k odběru témat v produktu IBM MQ se zasíláte publikacemi.

Kvality služby poskytované klientem MQTT
Klient produktu MQTT poskytuje tři úrovně kvality služby pro dodávání publikací do produktu IBM MQ a klientovi MQTT : "maximálně jednou", "alespoň jednou" a "přesně jednou". Když klient MQTT odešle požadavek produktu IBM MQ k vytvoření odběru, odešle se požadavek s kvalitou služby "alespoň jednou".

Zachovaná publikování a klienti MQTT
Téma může mít jednu, a pouze jednu zachovanou publikaci. Pokud vytvoříte odběr tématu, které má zachované publikování, bude vám ihned předáno publikování.

Odběry
Vytvoření odběrů pro registraci zájmu v tématech publikování pomocí filtru témat. Klient může vytvořit více odběrů nebo odběr obsahující filtr témat, který používá zástupné znaky, a zaregistrovat tak zájem o více témat. Publikace na témata, která odpovídají filtrům, se posílají na klienta. Odběry mohou zůstat aktivní, když je klient odpojen. Publikace se odesílají klientovi, jakmile se znovu připojí.

Řetězce témat a filtry témat v klientech produktu MQTT
Řetězce témat a filtry témat se používají pro publikování a odběr. Syntaxe řetězců témat a filtrů v klientech produktu MQTT je z velké části stejné jako řetězce témat v produktu IBM MQ.

Tokeny doručení

Když klient publikuje na téma nový token doručení, dojde k vytvoření nového tokenu doručení. Použijte token doručení k monitorování doručení publikování nebo k blokování klientské aplikace, dokud nebude doručení dokončeno.

Token je objekt `MqttDeliveryToken` . Vytvoří se voláním metody MqttTopic.publish () a zachová ji klient produktu MQTT , dokud nebude relace klienta odpojena a doručení je dokončeno.

Normální použití tokenu je zkontrolovat, zda je doručení dokončeno. Zablokujte aplikaci klienta, dokud se doručení nedokončí s použitím vráceného tokenu k volání `token.waitForCompletion`. Případně poskytněte obslužnou rutinu `MqttCallBack` . Pokud klient produktu MQTT přijal všechna potvrzení,

která očekává jako část doručení publikace, zavolá příkaz `MqttCallBack.deliveryComplete` předáním tokenu doručení jako parametru.

Dokud nebude dodávka dokončena, můžete ji zkontrolovat pomocí vráceného tokenu doručení voláním produktu `token.getMessage`.

Dokončené dodávky

Dokončení dodávek je asynchronní a závisí na kvalitě služby přidružené k publikování.

Nejvíce jednou

`QoS=0`
Doručení je dokončeno okamžitě po návratu z produktu `MqttTopic.publish`. `MqttCallback.deliveryComplete` se volá okamžitě.

Nejméně jednou

`QoS=1`
Doručení je dokončeno, když bylo od správce front přijato potvrzení o publikování. `MqttCallback.deliveryComplete` se volá, když je přijato potvrzení. Zpráva může být doručena více než jednou dříve, než se zavolá `MqttCallback.deliveryComplete` , pokud jsou komunikace pomalé nebo nespolehlivé.

Přesně jednou

`QoS=2`
Doručení je dokončeno, když klient obdrží zprávu o dokončení, že publikování bylo publikováno odběratelům. `MqttCallback.deliveryComplete` se volá, jakmile se přijme zpráva o publikování. Nečeká na zprávu o dokončení.

Za výjimečných okolností se nemusí klientská aplikace vrátit ke klientovi MQTT z produktu `MqttCallback.deliveryComplete` normálně. Víte, že dodávka byla dokončena, protože byl volán `MqttCallback.deliveryComplete` . Pokud klient restartuje stejnou relaci, produkt `MqttCallback.deliveryComplete` se znovu nezavolá.

Neúplné dodávky

Pokud není doručení dokončeno po odpojení relace klienta, můžete klienta znovu připojit a dokončit doručení. Doručování zprávy můžete dokončit pouze v případě, že byla zpráva publikována v relaci s atributem `MqttConnectionOptions` nastaveným na hodnotu `false`.

Vytvořte klienta pomocí stejného identifikátoru klienta a adresy serveru a potom se připojte znovu, nastavte atribut `cleanSession` `MqttConnectionOptions` na hodnotu `false` znovu. Pokud jste nastavili `cleanSession` na `true`, nevyřízené tokeny doručení budou zahozeny.

Můžete zkontrolovat, zda neexistují nějaké nevyřízené doručení voláním produktu `MqttClient.getPendingDeliveryTokens`. Před připojením klienta můžete volat `MqttClient.getPendingDeliveryTokens` .

Související pojmy

Zpětná volání a synchronizace v klientských aplikacích produktu MQTT
Programovací model klienta produktu MQTT rozsáhle používá podprocesy. Podprocesy oddělující aplikaci klienta MQTT , stejně jako mohou, z prodlev při přenosu zpráv na server a ze serveru. Publikování, tokeny doručení a ztracené události připojení se doručují do metod ve třídě zpětného volání, která implementuje `MqttCallback`.

Vyčistit relace
Klient produktu MQTT a služba telemetrie (MQXR) udržují informace o stavu relace. Informace o stavu se používají k zajištění doručení "alespoň jednou" a "přesně jednou" doručení a "přesně jednou" pro příjem publikací. Stav relace také zahrnuje odběry vytvořené klientem MQTT . Můžete zvolit spuštění klienta MQTT s nebo bez udržování informací o stavu mezi relacemi. Změňte režim čisté relace nastavením `MqttConnectOptions.cleanSession` před připojením.

Identifikátor klienta
Identifikátor klienta je 23bajtový řetězec, který identifikuje klienta MQTT . Každý identifikátor musí být jedinečný, tedy platný ve stejnou dobu pouze pro jediného připojeného klienta. Identifikátor musí obsahovat pouze znaky platné v názvu správce front. V rámci těchto omezení budete moci používat libovolný identifikační řetězec. Je důležité mít proceduru pro přidělování identifikátorů klientů a prostředků ke konfiguraci klienta s vybraným identifikátorem.

Datum poslední vůle a potvrzení
Dojde-li k neočekávanému ukončení připojení klienta MQTT , můžete produkt MQ Telemetry nakonfigurovat tak, aby odeslal publikaci "last will and testament". Předefinujte obsah publikace a téma, kterému jej chcete odeslat. "Poslední vůle a testament" je vlastnost připojení. Vytvořte jej před připojením klienta.

Perzistence zpráv v klientech produktu MQTT
Zprávy publikování jsou prováděny jako trvalé, pokud jsou odeslány s kvalitou služby "alespoň jednou", nebo "přesně jednou". Na klientovi můžete implementovat vlastní mechanismus perzistence nebo použít výchozí mechanismus perzistence, který je dodáván s klientem. Perzistence funguje v obou směrech, pro publikování odeslané na klienta nebo z klienta.

Publikace
Publikace jsou instance `MqttMessage` , které jsou přidruženy k řetězci tématu. Klienti produktu MQTT mohou vytvářet publikace k odesílání do produktu IBM MQa k odběru témat v produktu IBM MQ se zasíláte publikacemi.

Kvality služby poskytované klientem MQTT
Klient produktu MQTT poskytuje tři úrovně kvality služby pro dodávání publikací do produktu IBM MQ a klientovi MQTT : "maximálně jednou", "alespoň jednou" a "přesně jednou". Když klient MQTT odešle požadavek produktu IBM MQ k vytvoření odběru, odešle se požadavek s kvalitou služby "alespoň jednou".

Zachovaná publikování a klienti MQTT
Téma může mít jednu, a pouze jednu zachovanou publikaci. Pokud vytvoříte odběr tématu, které má zachované publikování, bude vám ihned předáno publikování.

Odběry
Vytvoření odběrů pro registraci zájmu v tématech publikování pomocí filtru témat. Klient může vytvořit více odběrů nebo odběr obsahující filtr témat, který používá zástupné znaky, a zaregistrovat tak zájem o více témat. Publikace na témata, která odpovídají filtrům, se posílají na klienta. Odběry mohou zůstat aktivní, když je klient odpojen. Publikace se odesílají klientovi, jakmile se znovu připojí.

Řetězce témat a filtry témat v klientech produktu MQTT
Řetězce témat a filtry témat se používají pro publikování a odběr. Syntaxe řetězců témat a filtrů v klientech produktu MQTT je z velké části stejné jako řetězce témat v produktu IBM MQ.

Datum poslední vůle a potvrzení

Dojde-li k neočekávanému ukončení připojení klienta MQTT , můžete produkt MQ Telemetry nakonfigurovat tak, aby odeslal publikaci "last will and testament". Předefinujte obsah publikace a téma, kterému jej chcete odeslat. "Poslední vůle a testament" je vlastnost připojení. Vytvořte jej před připojením klienta.

Vytvořte téma pro poslední vůli a testament. Můžete vytvořit téma jako např. `MQTTManagement/Connections/`*`server URI`*`/`*`client identifer`*`/Lost`.

Nastavte "poslední vůli a testament" pomocí metody `MqttConnectionOptions.setWill(MqttTopic lastWillTopic, byte [] lastWillPayload, int lastWillQos, boolean lastWillRetained)` .

Zvažte vytvoření časového razítka ve zprávě `lastWillPayload` . Zahrnout další informace o klientovi, které pomáhají při identifikaci klienta a okolnosti připojení. Předejte objekt `MqttConnectionOptions` konstruktoru `MqttClient` .

Nastavte parametr `lastWillQos` na hodnotu 1 nebo 2, aby byla zpráva perzistentní v produktu IBM MQa aby byla zaručena doručení. Chcete-li zachovat informace o naposledy ztraceném připojení, nastavte parametr `lastWillRetained` na hodnotu `true`.

Publikování "poslední vůle a testament" je odesláno odběratelům, pokud připojení skončí neočekávaně. Je odeslán, pokud připojení skončí, aniž by klient volal metodu `MqttClient.disconnect`.

Chcete-li monitorovat připojení, doplním publikace "last will and testament" s dalšími publikacemi k záznamu připojení a programovaným odpojením.

Související pojmy

Zpětná volání a synchronizace v klientských aplikacích produktu MQTT
Programovací model klienta produktu MQTT rozsáhle používá podprocesy. Podprocesy oddělující aplikaci klienta MQTT , stejně jako mohou, z prodlev při přenosu zpráv na server a ze serveru. Publikování, tokeny doručení a ztracené události připojení se doručují do metod ve třídě zpětného volání, která implementuje `MqttCallback`.

Vyčistit relace
Klient produktu MQTT a služba telemetrie (MQXR) udržují informace o stavu relace. Informace o stavu se používají k zajištění doručení "alespoň jednou" a "přesně jednou" doručení a "přesně jednou" pro příjem publikací. Stav relace také zahrnuje odběry vytvořené klientem MQTT . Můžete zvolit spuštění klienta MQTT s nebo bez udržování informací o stavu mezi relacemi. Změňte režim čisté relace nastavením `MqttConnectOptions.cleanSession` před připojením.

Identifikátor klienta
Identifikátor klienta je 23bajtový řetězec, který identifikuje klienta MQTT . Každý identifikátor musí být jedinečný, tedy platný ve stejnou dobu pouze pro jediného připojeného klienta. Identifikátor musí obsahovat pouze znaky platné v názvu správce front. V rámci těchto omezení budete moci používat libovolný identifikační řetězec. Je důležité mít proceduru pro přidělování identifikátorů klientů a prostředků ke konfiguraci klienta s vybraným identifikátorem.

Tokeny doručení

Perzistence zpráv v klientech produktu MQTT
Zprávy publikování jsou prováděny jako trvalé, pokud jsou odeslány s kvalitou služby "alespoň jednou", nebo "přesně jednou". Na klientovi můžete implementovat vlastní mechanismus perzistence nebo použít výchozí mechanismus perzistence, který je dodáván s klientem. Perzistence funguje v obou směrech, pro publikování odeslané na klienta nebo z klienta.

Publikace
Publikace jsou instance `MqttMessage` , které jsou přidruženy k řetězci tématu. Klienti produktu MQTT mohou vytvářet publikace k odesílání do produktu IBM MQa k odběru témat v produktu IBM MQ se zasíláte publikacemi.

Kvality služby poskytované klientem MQTT
Klient produktu MQTT poskytuje tři úrovně kvality služby pro dodávání publikací do produktu IBM MQ a klientovi MQTT : "maximálně jednou", "alespoň jednou" a "přesně jednou". Když klient MQTT odešle požadavek produktu IBM MQ k vytvoření odběru, odešle se požadavek s kvalitou služby "alespoň jednou".

Zachovaná publikování a klienti MQTT
Téma může mít jednu, a pouze jednu zachovanou publikaci. Pokud vytvoříte odběr tématu, které má zachované publikování, bude vám ihned předáno publikování.

Odběry
Vytvoření odběrů pro registraci zájmu v tématech publikování pomocí filtru témat. Klient může vytvořit více odběrů nebo odběr obsahující filtr témat, který používá zástupné znaky, a zaregistrovat tak zájem o více témat. Publikace na témata, která odpovídají filtrům, se posílají na klienta. Odběry mohou zůstat aktivní, když je klient odpojen. Publikace se odesílají klientovi, jakmile se znovu připojí.

Řetězce témat a filtry témat v klientech produktu MQTT
Řetězce témat a filtry témat se používají pro publikování a odběr. Syntaxe řetězců témat a filtrů v klientech produktu MQTT je z velké části stejné jako řetězce témat v produktu IBM MQ.

Perzistence zpráv v klientech produktu MQTT

Zprávy publikování jsou prováděny jako trvalé, pokud jsou odeslány s kvalitou služby "alespoň jednou", nebo "přesně jednou". Na klientovi můžete implementovat vlastní mechanismus perzistence nebo použít

výchozí mechanismus perzistence, který je dodáván s klientem. Perzistence funguje v obou směrech, pro publikování odeslané na klienta nebo z klienta.

V produktu MQTTmá perzistence zpráv dva aspekty, jak je zpráva přenášena a zda je zařazena do fronty v IBM MessageSight a IBM MQ jako trvalá zpráva.

1. Perzistence zpráv klientských párů MQTT s kvalitou služeb. V závislosti na tom, jakou kvalitu služby vyberete pro zprávu, bude zpráva trvalá. Perzistence zpráv je nezbytná k implementaci požadované kvality služby.

   Uvedete-li "nejvýše jednou", QoS=0, klient zprávu zahodí ihned, jakmile bude publikován. Dojde-li k selhání v předchozím zpracování zprávy, zpráva se neodešle znovu. I v případě, že klient zůstane aktivní, zpráva se znovu neodešle. Chování zpráv produktu QoS=0 se shoduje s chováním IBM MQ rychlých přechodných zpráv.

   Je-li zpráva publikována klientem s QoS z 1 nebo 2, je vytrvalá. Zpráva je uložena lokálně a pouze vyřazena z klienta, když již není zapotřebí zaručit "alespoň jednou", QoS=1nebo "přesně jednou", QoS=2, doručení.
2. Je-li zpráva označena jako QoS 1 nebo 2, je zařazena do fronty v IBM MessageSight a IBM MQ jako trvalá zpráva. Je-li označena jako QoS=0, pak je zařazena do fronty v IBM MessageSight a IBM MQ jako přechodná zpráva. V produktu IBM MQ jsou přechodné zprávy přenášeny mezi správci front "přesně jednou", pokud kanál zpráv nemá nastaven atribut NPMSPEED na hodnotu FAST.

Trvalá publikace je uložena v klientu, dokud ji nebude přijímat klientská aplikace. Pro produkt QoS=2je publikace vyřazena z klienta, když zpětné volání aplikace vrátí řízení. V případě QoS=1 může aplikace obdržet publikování znovu, pokud dojde k selhání. V případě systému QoS=0zpětné volání přijme publikování ne více než jednou. Pokud dojde k selhání nebo je-li klient odpojen v době publikování, nemusí být tato publikace zveřejněna.

Pokud se přihlásíte k odběru tématu, můžete snížit úroveň služeb QoS , se kterou odběratel přijímá zprávy, aby odpovídal schopnostem perzistence. Publikace, které jsou vytvořeny na vyšší verzi QoS , jsou odeslány s nejvyšší hodnotou QoS , kterou odběratel požadoval.

Ukládání zpráv

Implementace datového úložiště na malých zařízeních se velmi liší. Model dočasně ukládající trvalé zprávy v úložišti, který je spravován klientem MQTT , může být příliš pomalý, nebo požadovat příliš mnoho úložiště. V mobilních zařízeních může mobilní operační systém poskytovat paměťovou službu, která je ideální pro zprávy produktu MQTT .

Klient produktu MQTT má dvě rozhraní perzistence, aby byla zajištěna flexibilita při plnění omezení malých zařízení. Rozhraní definují operace, které jsou zapojeny do ukládání trvalých zpráv. Rozhraní jsou popsána v dokumentaci rozhraní API pro produkt Klient MQTT pro produkt Java. Pro odkazy na dokumentaci ke klientským rozhraním API pro knihovny klienta MQTT viz Odkaz na programování klienta MQTT. Rozhraní můžete implementovat tak, aby vyhovovala zařízení. Klient produktu MQTT spuštěný v produktu Java SE má výchozí implementaci rozhraní, která ukládají trvalé zprávy v systému souborů. Používá balík `java.io` .

Třídy perzistence

`MqttClientPersistence`
: Předejte instanci implementace produktu `MqttClientPersistence` na klienta produktu MQTT jako parametr konstruktoru `MqttClient` . Pokud vynecháte argument `MqttClientPersistence` z konstruktoru `MqttClient` , klient MQTT ukládá trvalé zprávy pomocí třídy `MqttDefaultFilePersistence`.

`MqttPersistable`
: `MqttClientPersistence` získává a vkládá objekty `MqttPersistable` pomocí klíče úložiště. Musíte poskytnout implementaci produktu `MqttPersistable` , stejně jako implementaci produktu `MqttClientPersistence` , pokud nepoužíváte `MqttDefaultFilePersistence`.

MqttDefaultFilePersistence

Klient MQTT poskytuje třídu `MqttDefaultFilePersistence` . Pokud konkretizovat `MqttDefaultFilePersistence` v aplikaci klienta, můžete poskytnout adresář k ukládání trvalých zpráv jako parametru konstruktoru `MqttDefaultFilePersistence` .

Alternativně může klient MQTT vytvořit instanci `MqttDefaultFilePersistence` a umístit soubory do následujícího výchozího adresáře:

`client identifier` `-tcp` *`hostname portnumber`*

Z řetězce názvu adresáře jsou odebrány následující znaky:

`"\", "\\", "/", ":" a " "`

Cesta k adresáři je hodnota vlastnosti systému `rcp.data`; Pokud `rcp.data` není nastaven, cesta je hodnotou systémové vlastnosti `usr.data`, kde

- `rcp.data` je vlastnost přidružená k instalaci platformy OSGi nebo platformy Eclipse Rich Client Platform (RCP).
- `usr.data` je adresář, ve kterém byl spuštěn příkaz Java , který spustil aplikaci.

Související pojmy

Zpětná volání a synchronizace v klientských aplikacích produktu MQTT
Programovací model klienta produktu MQTT rozsáhle používá podprocesy. Podprocesy oddělující aplikaci klienta MQTT , stejně jako mohou, z prodlev při přenosu zpráv na server a ze serveru. Publikování, tokeny doručení a ztracené události připojení se doručují do metod ve třídě zpětného volání, která implementuje `MqttCallback`.

Vyčistit relace
Klient produktu MQTT a služba telemetrie (MQXR) udržují informace o stavu relace. Informace o stavu se používají k zajištění doručení "alespoň jednou" a "přesně jednou" doručení a "přesně jednou" pro příjem publikací. Stav relace také zahrnuje odběry vytvořené klientem MQTT . Můžete zvolit spuštění klienta MQTT s nebo bez udržování informací o stavu mezi relacemi. Změňte režim čisté relace nastavením `MqttConnectOptions.cleanSession` před připojením.

Identifikátor klienta
Identifikátor klienta je 23bajtový řetězec, který identifikuje klienta MQTT . Každý identifikátor musí být jedinečný, tedy platný ve stejnou dobu pouze pro jediného připojeného klienta. Identifikátor musí obsahovat pouze znaky platné v názvu správce front. V rámci těchto omezení budete moci používat libovolný identifikační řetězec. Je důležité mít proceduru pro přidělování identifikátorů klientů a prostředků ke konfiguraci klienta s vybraným identifikátorem.

Tokeny doručení

Datum poslední vůle a potvrzení
Dojde-li k neočekávanému ukončení připojení klienta MQTT , můžete produkt MQ Telemetry nakonfigurovat tak, aby odeslal publikaci "last will and testament". Předefinujte obsah publikace a téma, kterému jej chcete odeslat. "Poslední vůle a testament" je vlastnost připojení. Vytvořte jej před připojením klienta.

Publikace
Publikace jsou instance `MqttMessage` , které jsou přidruženy k řetězci tématu. Klienti produktu MQTT mohou vytvářet publikace k odesílání do produktu IBM MQa k odběru témat v produktu IBM MQ se zasíláte publikacemi.

Kvality služby poskytované klientem MQTT
Klient produktu MQTT poskytuje tři úrovně kvality služby pro dodávání publikací do produktu IBM MQ a klientovi MQTT : "maximálně jednou", "alespoň jednou" a "přesně jednou". Když klient MQTT odešle požadavek produktu IBM MQ k vytvoření odběru, odešle se požadavek s kvalitou služby "alespoň jednou".

Zachovaná publikování a klienti MQTT
Téma může mít jednu, a pouze jednu zachovanou publikaci. Pokud vytvoříte odběr tématu, které má zachované publikování, bude vám ihned předáno publikování.

Odběry
Vytvoření odběrů pro registraci zájmu v tématech publikování pomocí filtru témat. Klient může vytvořit více odběrů nebo odběr obsahující filtr témat, který používá zástupné znaky, a zaregistrovat tak zájem o více témat. Publikace na témata, která odpovídají filtrům, se posílají na klienta. Odběry mohou zůstat aktivní, když je klient odpojen. Publikace se odesílají klientovi, jakmile se znovu připojí.

Řetězce témat a filtry témat v klientech produktu MQTT
Řetězce témat a filtry témat se používají pro publikování a odběr. Syntaxe řetězců témat a filtrů v klientech produktu MQTT je z velké části stejné jako řetězce témat v produktu IBM MQ.

Publikace

Publikace jsou instance `MqttMessage` , které jsou přidruženy k řetězci tématu. Klienti produktu MQTT mohou vytvářet publikace k odesílání do produktu IBM MQa k odběru témat v produktu IBM MQ se zasíláte publikacemi.

`MqttMessage` má jako svůj informační obsah pole bajtů. Zamířit, aby zprávy byly co nejmenší. Maximální délka zprávy povolená MQTT protocol je 250 MB.

Klientský program MQTT obvykle používá `java.lang.String` nebo `java.lang.StringBuffer` k manipulaci s obsahem zpráv. Pro usnadnění práce má třída `MqttMessage` metodu `toString` pro převod jejího informačního obsahu na řetězec. Chcete-li vytvořit informační obsah bajtového pole z `java.lang.String` nebo `java.lang.StringBuffer`, použijte metodu `getBytes` .

Metoda `getBytes` převádí řetězec na výchozí znakovou sadu pro platformu. Standardní znaková sada je obecně `UTF-8`. Příručky MQTT , které obsahují pouze text, jsou obvykle kódovány v produktu `UTF-8`. Chcete-li potlačit výchozí znakovou sadu, použijte metodu `getBytes("UTF8")` .

V produktu IBM MQje publikace MQTT přijata jako zpráva `jms-bytes` . Zpráva obsahuje složku `MQRFH2` obsahující složku `<mqtt>`a složku `<mqps>` . Složka `<mqtt>` obsahuje položky `clientId`, `msgId`a `qos`, ale tento obsah se může v budoucnu měnit.

`MqttMessage` má tři další atributy: kvalitu služby, ať už je uchován, a zda je duplikát. Duplicitní příznak je nastaven pouze v případě, že kvalita služby je "alespoň jednou", nebo "přesně jednou". Pokud byla zpráva poslána dříve a klientem MQTT nebyla dostatečně rychle potvrzena, zpráva se odešle znovu s duplicitním atributem nastaveným na `true`.

Publikování

Chcete-li vytvořit publikaci v aplikaci klienta MQTT , vytvořte `MqttMessage`. Nastavte informační obsah, kvalitu služby a informace o tom, zda je uchován, a volejte metodu `MqttTopic.publish(MqttMessage message)` , je vrácena `MqttDeliveryToken` a dokončení publikování je asynchronní.

Případně může klient produktu MQTT vytvořit dočasný objekt zprávy z parametrů v metodě `MqttTopic.publish(byte [] payload, int qos, boolean retained)` při vytváření publikace.

Pokud má publikování alespoň jednu úroveň kvality služeb `QoS=1` nebo `QoS=2`nebo "právě jednou", zavolá klient MQTT rozhraní `MqttClientPersistence` . Zavolá produkt `MqttClientPersistence` , aby uložil zprávu před vrácením doručovacího tokenu do aplikace.

Aplikace se může rozhodnout blokovat, dokud nebude zpráva doručena na server, pomocí metody `MqttDeliveryToken.waitForCompletion` . Alternativně může aplikace pokračovat bez blokování. Chcete-li zkontrolovat, zda jsou publikace doručeny, bez blokování, registrujte instanci třídy zpětného volání, která implementuje `MqttCallback` s klientem MQTT . Klient MQTT volá metodu `MqttCallback.deliveryComplete` , jakmile je publikace doručena. V závislosti na kvalitě služby může být doručení téměř okamžité pro `QoS=0`, nebo může nějakou dobu trvat, než `QoS=2`.

Pokud je doručení dokončeno, použijte metodu `MqttDeliveryToken.isComplete` . Zatímco hodnota `MqttDeliveryToken.isComplete` je `false`, můžete volat `MqttDeliveryToken.getMessage` pro získání obsahu zprávy. Pokud je výsledek volání `MqttDeliveryToken.isComplete` `true`, zpráva byla vyřazena a zavoláním příkazu `MqttDeliveryToken.getMessage` by došlo k výjimce ukazatele null. Mezi `MqttDeliveryToken.getMessage` a `MqttDeliveryToken.isCompleteneexistuje` žádná vestavěná synchronizace.

Pokud se klient odpojí před přijetím všech nevyřízených tokenů doručení, může se před připojením dotazovat nová instance klienta na nevyřízené tokeny doručení. Dokud se klient nepřipojí, nejsou dokončeny žádné nové dodávky a je bezpečné volat `MqttDeliveryToken.getMessage`. Chcete-li zjistit, které publikace nebyly doručeny, použijte metodu `MqttDeliveryToken.getMessage` . Nevyřízené tokeny doručení jsou zrušeny, pokud se připojíte k výchozí hodnotě `MqttConnectOptions.cleanSession` , `true`.

přihlášení odběru

Správce front nebo produkt IBM MessageSight je odpovědný za vytváření publikování pro odeslání na odběratele produktu MQTT . Správce front kontroluje, zda filtr témat v odběru vytvořeném klientem MQTT odpovídá řetězci tématu v publikování. Shoda může být buď přesná shoda, nebo může shoda obsahovat zástupné znaky. Před postoupením publikace odběrateli správce front zkontroluje správce front atributy tématu přidružené k této publikaci. Následuje postup vyhledávání popsaný v tématu Přihlášení k odběru pomocí řetězce tématu, který obsahuje zástupné znaky k identifikaci, zda objekt administrativního tématu uděluje oprávnění uživatele k odběru.

Když klient MQTT obdrží publikování s "alespoň jednou" kvalitou služby, volá metodu `MqttCallback.messageArrived` , aby zpracoval publikaci. Je-li kvalita služby publikace "přesně jednou", QoS=2klient MQTT volá rozhraní `MqttClientPersistence` , aby uložila zprávu při jejím přijetí. Pak zavolá příkaz `MqttCallback.messageArrived`.

Související pojmy

Zpětná volání a synchronizace v klientských aplikacích produktu MQTT
Programovací model klienta produktu MQTT rozsáhle používá podprocesy. Podprocesy oddělující aplikaci klienta MQTT , stejně jako mohou, z prodlev při přenosu zpráv na server a ze serveru. Publikování, tokeny doručení a ztracené události připojení se doručují do metod ve třídě zpětného volání, která implementuje `MqttCallback`.

Vyčistit relace
Klient produktu MQTT a služba telemetrie (MQXR) udržují informace o stavu relace. Informace o stavu se používají k zajištění doručení "alespoň jednou" a "přesně jednou" doručení a "přesně jednou" pro příjem publikací. Stav relace také zahrnuje odběry vytvořené klientem MQTT . Můžete zvolit spuštění klienta MQTT s nebo bez udržování informací o stavu mezi relacemi. Změňte režim čisté relace nastavením `MqttConnectOptions.cleanSession` před připojením.

Identifikátor klienta
Identifikátor klienta je 23bajtový řetězec, který identifikuje klienta MQTT . Každý identifikátor musí být jedinečný, tedy platný ve stejnou dobu pouze pro jediného připojeného klienta. Identifikátor musí obsahovat pouze znaky platné v názvu správce front. V rámci těchto omezení budete moci používat libovolný identifikační řetězec. Je důležité mít proceduru pro přidělování identifikátorů klientů a prostředků ke konfiguraci klienta s vybraným identifikátorem.

Tokeny doručení

Datum poslední vůle a potvrzení
Dojde-li k neočekávanému ukončení připojení klienta MQTT , můžete produkt MQ Telemetry nakonfigurovat tak, aby odeslal publikaci "last will and testament". Předefinujte obsah publikace a téma, kterému jej chcete odeslat. "Poslední vůle a testament" je vlastnost připojení. Vytvořte jej před připojením klienta.

Perzistence zpráv v klientech produktu MQTT
Zprávy publikování jsou prováděny jako trvalé, pokud jsou odeslány s kvalitou služby "alespoň jednou", nebo "přesně jednou". Na klientovi můžete implementovat vlastní mechanismus perzistence nebo použít výchozí mechanismus perzistence, který je dodáván s klientem. Perzistence funguje v obou směrech, pro publikování odeslané na klienta nebo z klienta.

Kvality služby poskytované klientem MQTT
Klient produktu MQTT poskytuje tři úrovně kvality služby pro dodávání publikací do produktu IBM MQ a klientovi MQTT : "maximálně jednou", "alespoň jednou" a "přesně jednou". Když klient MQTT odešle požadavek produktu IBM MQ k vytvoření odběru, odešle se požadavek s kvalitou služby "alespoň jednou".

Zachovaná publikování a klienti MQTT

Téma může mít jednu, a pouze jednu zachovanou publikaci. Pokud vytvoříte odběr tématu, které má zachované publikování, bude vám ihned předáno publikování.

Odběry
Vytvoření odběrů pro registraci zájmu v tématech publikování pomocí filtru témat. Klient může vytvořit více odběrů nebo odběr obsahující filtr témat, který používá zástupné znaky, a zaregistrovat tak zájem o více témat. Publikace na témata, která odpovídají filtrům, se posílají na klienta. Odběry mohou zůstat aktivní, když je klient odpojen. Publikace se odesílají klientovi, jakmile se znovu připojí.

Řetězce témat a filtry témat v klientech produktu MQTT
Řetězce témat a filtry témat se používají pro publikování a odběr. Syntaxe řetězců témat a filtrů v klientech produktu MQTT je z velké části stejné jako řetězce témat v produktu IBM MQ.

Kvality služby poskytované klientem MQTT

Klient produktu MQTT poskytuje tři úrovně kvality služby pro dodávání publikací do produktu IBM MQ a klientovi MQTT : "maximálně jednou", "alespoň jednou" a "přesně jednou". Když klient MQTT odešle požadavek produktu IBM MQ k vytvoření odběru, odešle se požadavek s kvalitou služby "alespoň jednou".

Quality of service of a publication is an attribute of `MqttMessage`. Je nastaven pomocí metody `MqttMessage.setQos`.

Metoda `MqttClient.subscribe` může snížit kvalitu služby aplikovanou na publikace odeslané klientovi na téma. Kvalita služeb publikace postoupené odběrateli se může lišit od kvality služby publikace. Nižší z těchto dvou hodnot se používá k předání publikace.

Nejvíce jednou
`QoS=0`

Zpráva je doručena nejvíce jednou nebo není doručena vůbec. Její doručení po síti není potvrzeno.
Zpráva není uložena. Při odpojení klienta nebo selhání serveru může dojít ke ztrátě zprávy.
`QoS=0` je nejrychlejší způsob přenosu. Někdy se říká "oheň a zapomenout".
Produkt MQTT protocol nevyžaduje od serverů předávání publikací v produktu `QoS=0` klientovi. Pokud je klient odpojen v době, kdy server obdrží publikování, může být publikování zrušeno, v závislosti na serveru. Služba telemetrie (MQXR) nevyřazovat zprávy odeslané s produktem `QoS=0`. Jsou uloženy jako přechodné zprávy a jsou zahozeny pouze v případě, že je správce front zastaven.

Nejméně jednou
`QoS=1`

`QoS=1` je výchozí režim přenosu.
Zpráva se vždy doručí alespoň jednou. Pokud odesilatel neobdrží potvrzení, je zpráva odeslána znovu s příznakem `DUP` , dokud není přijato potvrzení. V důsledku toho lze příjemce odeslat stejnou zprávu vícekrát a může jej zpracovávat vícekrát.
Zpráva musí být uložena lokálně na odesílateli a přijímači, dokud nebude zpracována.
Zpráva se odstraní z přijímače poté, co zpracoval zprávu. Je-li příjemcem zprostředkovatel, zpráva se publikuje na své odběratele. Je-li příjemcem klient, zpráva se doručí do aplikace odběratele. Jakmile je zpráva odstraněna, příjemce odešle potvrzení odesilateli.
Zpráva se odstraní od odesílatele poté, co mu bylo přijato potvrzení od příjemce.

Přesně jednou
`QoS=2`

Zpráva se vždy doručí přesně jednou.
Zpráva musí být uložena lokálně na odesílateli a přijímači, dokud nebude zpracována.
`QoS=2` je nejbezpečnější, ale nejpomalejší způsob přenosu. Mezi odesilatelem a příjemcem je třeba provést alespoň dva páry přenosů před tím, než je zpráva odstraněna od odesílatele. Zprávu lze na příjemce zpracovat po prvním přenosu.
V první dvojici přenosů odešle odesilatel zprávu a získá potvrzení od příjemce, že uložil zprávu. Pokud odesilatel neobdrží potvrzení, je zpráva odeslána znovu s příznakem `DUP` , dokud není přijato potvrzení.

Ve druhém páru přenosů odesilatel sděluje přijímači, že může dokončit zpracování zprávy, "PUBREL". Pokud odesilatel neobdrží potvrzení o zprávě "PUBREL" , je zpráva "PUBREL" poslána znovu, dokud neobdrží potvrzení. Odesilatel odstraní zprávu, která byla uložena, když přijme potvrzení pro zprávu "PUBREL" .

Příjemce může zprávu zpracovat v první nebo druhé fázi za předpokladu, že tuto zprávu nezpracuje znovu. Je-li příjemcem zprostředkovatel, publikuje zprávu pro odběratele. Je-li příjemcem klient, doručí zprávu do aplikace odběratele. Příjemce pošle zprávu o dokončení zpět odesílateli, že dokončil zpracování zprávy.

Související pojmy

Zpětná volání a synchronizace v klientských aplikacích produktu MQTT
Programovací model klienta produktu MQTT rozsáhle používá podprocesy. Podprocesy oddělující aplikaci klienta MQTT , stejně jako mohou, z prodlev při přenosu zpráv na server a ze serveru. Publikování, tokeny doručení a ztracené události připojení se doručují do metod ve třídě zpětného volání, která implementuje `MqttCallback`.

Vyčistit relace
Klient produktu MQTT a služba telemetrie (MQXR) udržují informace o stavu relace. Informace o stavu se používají k zajištění doručení "alespoň jednou" a "přesně jednou" doručení a "přesně jednou" pro příjem publikací. Stav relace také zahrnuje odběry vytvořené klientem MQTT . Můžete zvolit spuštění klienta MQTT s nebo bez udržování informací o stavu mezi relacemi. Změňte režim čisté relace nastavením `MqttConnectOptions.cleanSession` před připojením.

Identifikátor klienta
Identifikátor klienta je 23bajtový řetězec, který identifikuje klienta MQTT . Každý identifikátor musí být jedinečný, tedy platný ve stejnou dobu pouze pro jediného připojeného klienta. Identifikátor musí obsahovat pouze znaky platné v názvu správce front. V rámci těchto omezení budete moci používat libovolný identifikační řetězec. Je důležité mít proceduru pro přidělování identifikátorů klientů a prostředků ke konfiguraci klienta s vybraným identifikátorem.

Tokeny doručení

Datum poslední vůle a potvrzení
Dojde-li k neočekávanému ukončení připojení klienta MQTT , můžete produkt MQ Telemetry nakonfigurovat tak, aby odeslal publikaci "last will and testament". Předefinujte obsah publikace a téma, kterému jej chcete odeslat. "Poslední vůle a testament" je vlastnost připojení. Vytvořte jej před připojením klienta.

Perzistence zpráv v klientech produktu MQTT
Zprávy publikování jsou prováděny jako trvalé, pokud jsou odeslány s kvalitou služby "alespoň jednou", nebo "přesně jednou". Na klientovi můžete implementovat vlastní mechanismus perzistence nebo použít výchozí mechanismus perzistence, který je dodáván s klientem. Perzistence funguje v obou směrech, pro publikování odeslané na klienta nebo z klienta.

Publikace
Publikace jsou instance `MqttMessage` , které jsou přidruženy k řetězci tématu. Klienti produktu MQTT mohou vytvářet publikace k odesílání do produktu IBM MQa k odběru témat v produktu IBM MQ se zasíláte publikacemi.

Zachovaná publikování a klienti MQTT
Téma může mít jednu, a pouze jednu zachovanou publikaci. Pokud vytvoříte odběr tématu, které má zachované publikování, bude vám ihned předáno publikování.

Odběry
Vytvoření odběrů pro registraci zájmu v tématech publikování pomocí filtru témat. Klient může vytvořit více odběrů nebo odběr obsahující filtr témat, který používá zástupné znaky, a zaregistrovat tak zájem o více témat. Publikace na témata, která odpovídají filtrům, se posílají na klienta. Odběry mohou zůstat aktivní, když je klient odpojen. Publikace se odesílají klientovi, jakmile se znovu připojí.

Řetězce témat a filtry témat v klientech produktu MQTT
Řetězce témat a filtry témat se používají pro publikování a odběr. Syntaxe řetězců témat a filtrů v klientech produktu MQTT je z velké části stejné jako řetězce témat v produktu IBM MQ.

Zachovaná publikování a klienti MQTT

Téma může mít jednu, a pouze jednu zachovanou publikaci. Pokud vytvoříte odběr tématu, které má zachované publikování, bude vám ihned předáno publikování.

Pomocí metody `MqttMessage.setRetained` lze určit, zda má být publikování na téma zachováno.

Když vytvoříte nebo aktualizujete zachované publikování, odešlete publikaci se `QoS` z 1 nebo 2. Pokud ji odešlete pomocí `QoS` z 0, produkt IBM MQ vytvoří netrvalé zachované publikování. Publikování nebude zachováno, je-li správce front zastaven.

Pokud publikujete nezachované publikování na téma, které má zachované publikování, zachované publikování nebude mít vliv na zachované publikování. Aktuální odběratelé obdrží novou publikaci. Noví odběratelé obdrží zachované publikování jako první a poté obdrží nové publikace.

Zachované publikování můžete použít k zaznamenání nejnovější hodnoty měření. Noví odběratelé na téma okamžitě obdrží nejnovější hodnotu měření. Pokud od odběratele, který byl naposledy přihlášen k tématu publikování, nebyla provedena žádná nová měření, a pokud se odběratel znovu přihlásí, odběratel obdrží nejnovější zachované publikování v rámci tématu znovu.

Chcete-li odstranit zachované publikování, máte dvě možnosti:

- Spusťte příkaz MQSC **`CLEAR TOPICSTR`** .
- Vytvořte zachované publikování s nulovou délkou. Jak je uvedeno ve specifikaci MQTT 3.1.1 , pokud je uchovaná zpráva s nulovou délkou publikována na téma, veškerá zachovaná zpráva pro dané téma se vymaže.

Související pojmy

Zpětná volání a synchronizace v klientských aplikacích produktu MQTT
Programovací model klienta produktu MQTT rozsáhle používá podprocesy. Podprocesy oddělující aplikaci klienta MQTT , stejně jako mohou, z prodlev při přenosu zpráv na server a ze serveru. Publikování, tokeny doručení a ztracené události připojení se doručují do metod ve třídě zpětného volání, která implementuje `MqttCallback`.

Vyčistit relace
Klient produktu MQTT a služba telemetrie (MQXR) udržují informace o stavu relace. Informace o stavu se používají k zajištění doručení "alespoň jednou" a "přesně jednou" doručení a "přesně jednou" pro příjem publikací. Stav relace také zahrnuje odběry vytvořené klientem MQTT . Můžete zvolit spuštění klienta MQTT s nebo bez udržování informací o stavu mezi relacemi. Změňte režim čisté relace nastavením `MqttConnectOptions.cleanSession` před připojením.

Identifikátor klienta
Identifikátor klienta je 23bajtový řetězec, který identifikuje klienta MQTT . Každý identifikátor musí být jedinečný, tedy platný ve stejnou dobu pouze pro jediného připojeného klienta. Identifikátor musí obsahovat pouze znaky platné v názvu správce front. V rámci těchto omezení budete moci používat libovolný identifikační řetězec. Je důležité mít proceduru pro přidělování identifikátorů klientů a prostředků ke konfiguraci klienta s vybraným identifikátorem.

Tokeny doručení

Datum poslední vůle a potvrzení
Dojde-li k neočekávanému ukončení připojení klienta MQTT , můžete produkt MQ Telemetry nakonfigurovat tak, aby odeslal publikaci "last will and testament". Předefinujte obsah publikace a téma, kterému jej chcete odeslat. "Poslední vůle a testament" je vlastnost připojení. Vytvořte jej před připojením klienta.

Perzistence zpráv v klientech produktu MQTT
Zprávy publikování jsou prováděny jako trvalé, pokud jsou odeslány s kvalitou služby "alespoň jednou", nebo "přesně jednou". Na klientovi můžete implementovat vlastní mechanismus perzistence nebo použít výchozí mechanismus perzistence, který je dodáván s klientem. Perzistence funguje v obou směrech, pro publikování odeslané na klienta nebo z klienta.

Publikace

Publikace jsou instance `MqttMessage` , které jsou přidruženy k řetězci tématu. Klienti produktu MQTT mohou vytvářet publikace k odesílání do produktu IBM MQa k odběru témat v produktu IBM MQ se zasíláte publikacemi.

Kvality služby poskytované klientem MQTT
Klient produktu MQTT poskytuje tři úrovně kvality služby pro dodávání publikací do produktu IBM MQ a klientovi MQTT : "maximálně jednou", "alespoň jednou" a "přesně jednou". Když klient MQTT odešle požadavek produktu IBM MQ k vytvoření odběru, odešle se požadavek s kvalitou služby "alespoň jednou".

Odběry
Vytvoření odběrů pro registraci zájmu v tématech publikování pomocí filtru témat. Klient může vytvořit více odběrů nebo odběr obsahující filtr témat, který používá zástupné znaky, a zaregistrovat tak zájem o více témat. Publikace na témata, která odpovídají filtrům, se posílají na klienta. Odběry mohou zůstat aktivní, když je klient odpojen. Publikace se odesílají klientovi, jakmile se znovu připojí.

Řetězce témat a filtry témat v klientech produktu MQTT
Řetězce témat a filtry témat se používají pro publikování a odběr. Syntaxe řetězců témat a filtrů v klientech produktu MQTT je z velké části stejné jako řetězce témat v produktu IBM MQ.

Odběry

Vytvoření odběrů pro registraci zájmu v tématech publikování pomocí filtru témat. Klient může vytvořit více odběrů nebo odběr obsahující filtr témat, který používá zástupné znaky, a zaregistrovat tak zájem o více témat. Publikace na témata, která odpovídají filtrům, se posílají na klienta. Odběry mohou zůstat aktivní, když je klient odpojen. Publikace se odesílají klientovi, jakmile se znovu připojí.

Vytvořte odběry pomocí metod `MqttClient.subscribe` , které jsou předáním jednoho nebo více filtrů témat a parametrů kvality služby. Parametr quality of service nastavuje maximální kvalitu služby, kterou je odběratel připraven použít pro příjem zprávy. Zprávy odeslané tomuto klientovi nemohou být doručeny s vyšší kvalitou služby. Kvalita služby je nastavena na nižší z původní hodnoty, když byla zpráva publikována a úroveň uvedená pro odběr. Výchozí kvalita služby pro příjem zpráv je `QoS=1`, alespoň jednou.

Samotný požadavek na odběr je odeslán s produktem `QoS=1`.

Publications přijímá odběratel, když klient MQTT volá metodu `MqttCallback.messageArrived` . Metoda `messageArrived` také předává řetězec tématu, se kterým byla zpráva publikována, na odběratele.

Pomocí metod `MqttClient.unsubscribe` můžete odebrat odběr nebo sadu či odběry.

Příkazu IBM MQ lze odebrat odběr. Seznam odběrů pomocí produktu IBM MQ Explorernebo pomocí příkazů **runmqsc** nebo PCF. Všechny odběry klienta MQTT mají název. Zobrazí se název formuláře: *`ClientIdentifier`*:*`Topic name`*

Pokud použijete výchozí `MqttConnectOptions`nebo nastavíte `MqttConnectOptions.cleanSession` na `true` před připojením klienta, všechny staré odběry klienta se odeberou, když se klient připojí. Všechny nové odběry, které klient během relace vytvoří, jsou při odpojení odebrány.

Pokud před připojením nastavíte `MqttConnectOptions.cleanSession` na `false` , všechny odběry vytvořené klientem budou přidány ke všem odběrům, které existovaly pro klienta dříve, než se připojí. Všechny odběry zůstávají aktivní, když se klient odpojí.

Dalším přístupem k pochopení, jak atribut `cleanSession` ovlivňuje odběry, je považovat jej za modální atribut. Ve svém výchozím režimu, `cleanSession=true`, vytváří klient odběry a přijímá publikace pouze v rámci dané relace. V alternativním režimu, `cleanSession=false`, jsou odběry trvalé. Klient se může připojit a odpojit a jeho odběry zůstanou aktivní. Když se klient znovu připojí, přijme všechny nedoručené publikace. V době připojení může upravit sadu odběrů, které jsou jeho jménem aktivní.

Před připojením musíte nastavit režim `cleanSession` ; režim trvá pro celou relaci. Chcete-li změnit jeho nastavení, je třeba klienta odpojit a znovu připojit. Pokud změníte režimy z použití `cleanSession=false` na `cleanSession=true`, všechny předchozí odběry pro klienta a všechny publikace, které nebyly přijaty, budou vyřazeny.

Publikace, které odpovídají aktivním odběrům, se odešlou klientovi, jakmile jsou publikovány. Je-li klient odpojen, odešle se klientovi, pokud se znovu připojí ke stejnému serveru se stejným identifikátorem klienta a `MqttConnectOptions.cleanSession` nastaveným na `false`.

Odběry pro určitého klienta jsou identifikovány identifikátorem klienta. Klienta můžete znovu připojit z jiného klientského zařízení na stejný server a pokračovat se stejnými odběry a přijímat nedoručené publikace.

Související pojmy

Zpětná volání a synchronizace v klientských aplikacích produktu MQTT
Programovací model klienta produktu MQTT rozsáhle používá podprocesy. Podprocesy oddělující aplikaci klienta MQTT , stejně jako mohou, z prodlev při přenosu zpráv na server a ze serveru. Publikování, tokeny doručení a ztracené události připojení se doručují do metod ve třídě zpětného volání, která implementuje `MqttCallback`.

Vyčistit relace
Klient produktu MQTT a služba telemetrie (MQXR) udržují informace o stavu relace. Informace o stavu se používají k zajištění doručení "alespoň jednou" a "přesně jednou" doručení a "přesně jednou" pro příjem publikací. Stav relace také zahrnuje odběry vytvořené klientem MQTT . Můžete zvolit spuštění klienta MQTT s nebo bez udržování informací o stavu mezi relacemi. Změňte režim čisté relace nastavením `MqttConnectOptions.cleanSession` před připojením.

Identifikátor klienta
Identifikátor klienta je 23bajtový řetězec, který identifikuje klienta MQTT . Každý identifikátor musí být jedinečný, tedy platný ve stejnou dobu pouze pro jediného připojeného klienta. Identifikátor musí obsahovat pouze znaky platné v názvu správce front. V rámci těchto omezení budete moci používat libovolný identifikační řetězec. Je důležité mít proceduru pro přidělování identifikátorů klientů a prostředků ke konfiguraci klienta s vybraným identifikátorem.

Tokeny doručení

Datum poslední vůle a potvrzení
Dojde-li k neočekávanému ukončení připojení klienta MQTT , můžete produkt MQ Telemetry nakonfigurovat tak, aby odeslal publikaci "last will and testament". Předefinujte obsah publikace a téma, kterému jej chcete odeslat. "Poslední vůle a testament" je vlastnost připojení. Vytvořte jej před připojením klienta.

Perzistence zpráv v klientech produktu MQTT
Zprávy publikování jsou prováděny jako trvalé, pokud jsou odeslány s kvalitou služby "alespoň jednou", nebo "přesně jednou". Na klientovi můžete implementovat vlastní mechanismus perzistence nebo použít výchozí mechanismus perzistence, který je dodáván s klientem. Perzistence funguje v obou směrech, pro publikování odeslané na klienta nebo z klienta.

Publikace
Publikace jsou instance `MqttMessage` , které jsou přidruženy k řetězci tématu. Klienti produktu MQTT mohou vytvářet publikace k odesílání do produktu IBM MQa k odběru témat v produktu IBM MQ se zasíláte publikacemi.

Kvality služby poskytované klientem MQTT
Klient produktu MQTT poskytuje tři úrovně kvality služby pro dodávání publikací do produktu IBM MQ a klientovi MQTT : "maximálně jednou", "alespoň jednou" a "přesně jednou". Když klient MQTT odešle požadavek produktu IBM MQ k vytvoření odběru, odešle se požadavek s kvalitou služby "alespoň jednou".

Zachovaná publikování a klienti MQTT
Téma může mít jednu, a pouze jednu zachovanou publikaci. Pokud vytvoříte odběr tématu, které má zachované publikování, bude vám ihned předáno publikování.

Řetězce témat a filtry témat v klientech produktu MQTT
Řetězce témat a filtry témat se používají pro publikování a odběr. Syntaxe řetězců témat a filtrů v klientech produktu MQTT je z velké části stejné jako řetězce témat v produktu IBM MQ.

Řetězce témat a filtry témat v klientech produktu MQTT

Řetězce témat a filtry témat se používají pro publikování a odběr. Syntaxe řetězců témat a filtrů v klientech produktu MQTT je z velké části stejné jako řetězce témat v produktu IBM MQ.

Řetězce témat se používají k odeslání publikování na odběratele. Vytvořte řetězec tématu za použití metody `MqttClient.getTopic(java.lang.String topicString)`.

Filtry témat se používají k odběru témat a přijímat publikování. Filtry témat mohou obsahovat zástupné znaky. Se zástupnými znaky se můžete přihlásit k odběru více témat. Vytvořte filtr témat pomocí metody odběru; například `MqttClient.subscribe(java.lang.String topicFilter)`.

Řetězce tématu

Syntaxe řetězce tématu IBM MQ je popsána v tématu Řetězce tématu. Syntaxe řetězců témat MQTT je popsána ve třídě `MqttClient` v dokumentaci k rozhraním API pro Klient MQTT pro produkt Java. Pro odkazy na dokumentaci ke klientským rozhraním API pro knihovny klienta MQTT viz Odkaz na programování klienta MQTT.

Syntaxe jednotlivých typů řetězců témat je téměř identická. Jsou zde čtyři menší rozdíly:

1. Topic strings sent to IBM MQ by MQTT clients must follow the convention for queue manager names.
2. Maximální délka se liší. Řetězce témat IBM MQ jsou omezeny na 10,240 znaků. Klient produktu MQTT může vytvořit řetězce témat až 65535 bajtů.
3. Řetězec tématu vytvořený klientem MQTT nemůže obsahovat znak null.
4. V produktu IBM Integration Busje úroveň tématu null, `'...//...'` je neplatná. Hodnoty témat s hodnotou null jsou podporovány produktem IBM MQ.

Na rozdíl od IBM MQ publish/subscribe, `mqttv3` protokol nemá koncepci objektu administrativního tématu. Řetězec tématu nelze zkonstruovat z objektu tématu a z řetězce tématu. Řetězec tématu je však mapován na administrativní téma v produktu IBM MQ. Řízení přístupu přidružené k administrativnímu tématu určuje, zda je publikování publikováno do tématu, nebo zrušeno. Atributy, které jsou použity ke zveřejnění při jeho předání odběratelům, jsou ovlivněny atributy administrativního tématu.

Filtry témat

Syntaxe filtru témat IBM MQ je popsána v tématu Schéma zástupných znaků na základě témat. Syntaxe filtrů témat, které lze sestavit s klientem MQTT , je popsána ve třídě `MqttClient` v dokumentaci k rozhraním API pro Klient MQTT pro produkt Java. Pro odkazy na dokumentaci ke klientským rozhraním API pro knihovny klienta MQTT viz Odkaz na programování klienta MQTT.

Související pojmy

Zpětná volání a synchronizace v klientských aplikacích produktu MQTT
Programovací model klienta produktu MQTT rozsáhle používá podprocesy. Podprocesy oddělující aplikaci klienta MQTT , stejně jako mohou, z prodlev při přenosu zpráv na server a ze serveru. Publikování, tokeny doručení a ztracené události připojení se doručují do metod ve třídě zpětného volání, která implementuje `MqttCallback`.

Vyčistit relace
Klient produktu MQTT a služba telemetrie (MQXR) udržují informace o stavu relace. Informace o stavu se používají k zajištění doručení "alespoň jednou" a "přesně jednou" doručení a "přesně jednou" pro příjem publikací. Stav relace také zahrnuje odběry vytvořené klientem MQTT . Můžete zvolit spuštění klienta MQTT s nebo bez udržování informací o stavu mezi relacemi. Změňte režim čisté relace nastavením `MqttConnectOptions.cleanSession` před připojením.

Identifikátor klienta
Identifikátor klienta je 23bajtový řetězec, který identifikuje klienta MQTT . Každý identifikátor musí být jedinečný, tedy platný ve stejnou dobu pouze pro jediného připojeného klienta. Identifikátor musí obsahovat pouze znaky platné v názvu správce front. V rámci těchto omezení budete moci používat libovolný identifikační řetězec. Je důležité mít proceduru pro přidělování identifikátorů klientů a prostředků ke konfiguraci klienta s vybraným identifikátorem.

Tokeny doručení

Datum poslední vůle a potvrzení
Dojde-li k neočekávanému ukončení připojení klienta MQTT , můžete produkt MQ Telemetry nakonfigurovat tak, aby odeslal publikaci "last will and testament". Předefinujte obsah publikace a téma, kterému jej chcete odeslat. "Poslední vůle a testament" je vlastnost připojení. Vytvořte jej před připojením klienta.

Perzistence zpráv v klientech produktu MQTT
Zprávy publikování jsou prováděny jako trvalé, pokud jsou odeslány s kvalitou služby "alespoň jednou", nebo "přesně jednou". Na klientovi můžete implementovat vlastní mechanismus perzistence nebo použít výchozí mechanismus perzistence, který je dodáván s klientem. Perzistence funguje v obou směrech, pro publikování odeslané na klienta nebo z klienta.

Publikace
Publikace jsou instance `MqttMessage` , které jsou přidruženy k řetězci tématu. Klienti produktu MQTT mohou vytvářet publikace k odesílání do produktu IBM MQa k odběru témat v produktu IBM MQ se zasíláte publikacemi.

Kvality služby poskytované klientem MQTT
Klient produktu MQTT poskytuje tři úrovně kvality služby pro dodávání publikací do produktu IBM MQ a klientovi MQTT : "maximálně jednou", "alespoň jednou" a "přesně jednou". Když klient MQTT odešle požadavek produktu IBM MQ k vytvoření odběru, odešle se požadavek s kvalitou služby "alespoň jednou".

Zachovaná publikování a klienti MQTT
Téma může mít jednu, a pouze jednu zachovanou publikaci. Pokud vytvoříte odběr tématu, které má zachované publikování, bude vám ihned předáno publikování.

Odběry
Vytvoření odběrů pro registraci zájmu v tématech publikování pomocí filtru témat. Klient může vytvořit více odběrů nebo odběr obsahující filtr témat, který používá zástupné znaky, a zaregistrovat tak zájem o více témat. Publikace na témata, která odpovídají filtrům, se posílají na klienta. Odběry mohou zůstat aktivní, když je klient odpojen. Publikace se odesílají klientovi, jakmile se znovu připojí.

Vývoj aplikací Microsoft Windows Communication Foundation (WCF) s IBM MQ

Vlastní kanál Microsoft Windows Communication Foundation (WCF) pro produkt IBM MQ odesílá a přijímá zprávy mezi klienty WCF a službami.

Související pojmy

“Úvod do použití vlastního kanálu produktu IBM MQ pro prostředí WCF s produktem .NET verze 3” na stránce 1218
Přehled informací dostupných pro programátory pomocí vlastního kanálu produktu IBM MQ pro produkt Windows Communication Foundation (WCF) s .NET 3.

“Použití vlastních kanálů produktu IBM MQ pro prostředek WCF” na stránce 1223
Přehled informací dostupných pro programátory používající IBM MQ vlastní kanály pro produkt Windows Communication Foundation (WCF).

“Použití ukázek WCF” na stránce 1242
Ukázky produktu Windows Communication Foundation (WCF) poskytují některé jednoduché příklady použití vlastního kanálu produktu IBM MQ .

“Určování problémů s vlastním kanálem WCF pro produkt IBM MQ” na stránce 1248
Pomocí trasování produktu IBM MQ můžete shromažďovat podrobné informace o tom, které různé části kódu IBM MQ dělají. Při použití produktu Windows Communication Foundation (WCF) je generován samostatný výstup trasování pro trasování vlastního kanálu WCF, který je integrován s trasováním infrastruktury WCF produktu Microsoft .

Úvod do použití vlastního kanálu produktu IBM MQ pro prostředí WCF s produktem .NET verze 3

Přehled informací dostupných pro programátory pomocí vlastního kanálu produktu IBM MQ pro produkt Windows Communication Foundation (WCF) s .NET 3.

Co je vlastní kanál IBM MQ pro WCF?

Vlastní kanál pro produkt IBM MQ je transportním kanálem s jednotným programovacím modelem produktu Microsoft Windows Communication Foundation (WCF).

Rámec produktu Microsoft Windows Communication Foundation, představený v produktu Microsoft.NET 3, umožňuje vývoj aplikací a služeb produktu .NET nezávisle na přenosu a protokolech použitých pro jejich připojení, povolení alternativních přenosů nebo konfigurací, které mají být použity v souladu s prostředím, v němž je služba nebo aplikace implementována.

Produkt WCF je spravován systémem Connections tak, že vytvoří zásobník kanálu obsahující požadovanou kombinaci:

- Protokolové prvky: Nepovinná sada prvků, kde žádný, jeden nebo více lze přidat do podpůrných protokolů, jako jsou standardy WS-*.
- Kodér zprávy: Povinný prvek v zásobníku, který řídí serializaci zprávy do svého formátu spoje.
- Transportní kanál: Povinný prvek v sadě, který je zodpovědný za přenos serializovaných zpráv do koncového bodu.

Vlastní kanál pro produkt IBM MQ je přenosový kanál a jako takový musí být spárován s kódovacím programem zpráv a s volitelnými protokoly, jak to vyžaduje aplikace používající vlastní vazbu WCF. Tímto způsobem aplikace, které byly vyvinuty pro použití WCF, mohou používat vlastní kanál pro produkt IBM MQ k odesílání a přijímání dat stejným způsobem, jako používají vestavěné přenosy poskytované produktem Microsofta umožňují jednoduchou integraci s funkcemi asynchronního, rozšiřitelného a spolehlivého systému zpráv produktu IBM MQ. Úplný seznam podporovaných funkcí viz: “Funkce a schopnosti vlastního kanálu služby WCF” na stránce 1223.

Kdy a proč používat vlastní kanál IBM MQ pro WCF?

Pomocí vlastního kanálu produktu IBM MQ můžete odesílat a přijímat zprávy mezi klienty WCF a službami stejným způsobem, jako jsou vestavěné přenosy poskytované produktem Microsoft, což umožňuje aplikacím přístup k funkcím produktu IBM MQ v rámci unifikovaného programovacího modelu WCF.

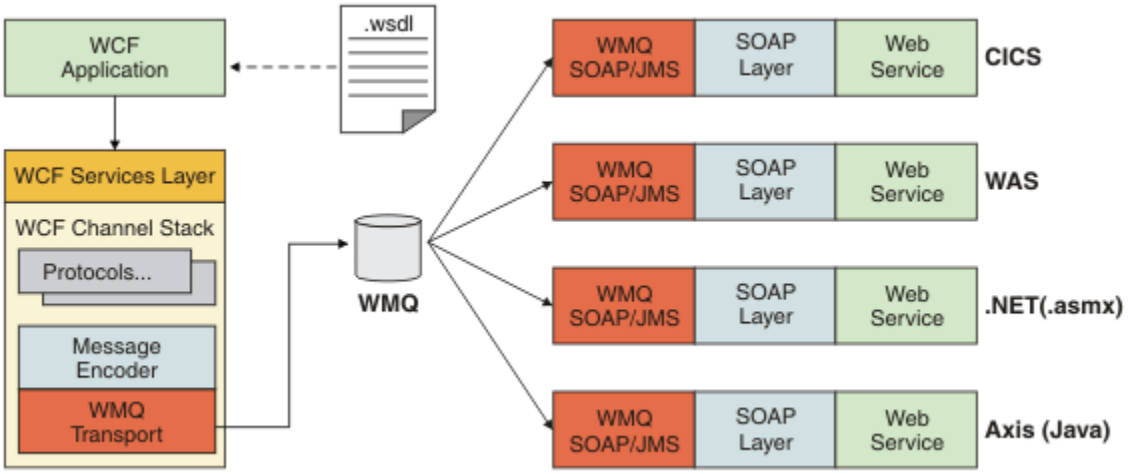
Typické scénáře použití pro vlastní kanál produktu IBM MQ pro prostředek WCF jsou:

- Jako rozhraní pro webové služby hostované přes IBM MQ (SOAP přes JMS).
- Jako rozhraní jiného typu než SOAP pro přenos nativních zpráv produktu IBM MQ .

Zprávy přenesené pomocí formátu protokolu SOAP prostřednictvím produktu JMS

Typický scénář vzorku použití vlastního kanálu produktu IBM MQ pro WCF je jako rozhraní webových služeb hostovaných nad produktem IBM MQ (SOAP/JMS).

Messages are carried using the SOAP over JMS message format of IBM MQ, enabling WCF clients and services to also call or be called by other WebSphere Application Server applications or hosting environments which are compatible with this format, including web services and clients running in , CICS, Axis v1 (Java), and .asmx (.NET), as shown in the following diagram:



Podrobné informace o protokolu SOAP prostřednictvím produktu JMSviz: "Vyvíjení webových služeb s přenosem produktu IBM MQ pro SOAP" na stránce 1256

Příklad typického scénáře z diagramu by byl:

1. Webová služba provozovaná v rámci produktu WebSphere Application Server a vystavená prostřednictvím produktu IBM MQ za použití podpory protokolu SOAP prostřednictvím produktu JMS v rámci produktu WebSphere Application Server.
2. Dokument WSDL popisující službu pak může nástroj WCF použít k vygenerování serveru proxy klienta a konfigurace, který by pak vytvořil příslušnou sadu kanálů WCF včetně vlastního kanálu.
3. Klientská aplikace pak může použít server proxy ke spuštění webové služby stejným způsobem jako jakákoli jiná webová služba.

Kanál by se obvykle použil s kódováním zpráv WCF text/SOAP, ale kanál může být v případě potřeby spárován s jinými enkodéry zpráv WCF. Použití alternativního kódovače může také poskytnout omezenou integraci s nativními aplikacemi IBM MQ , které nepodporují protokol SOAP přes JMS, ale to není primární role kanálu.

Mezi klíčové výhody použití vlastního kanálu v prostředí WCF patří:

- Asynchronní vyvolání: Podpora operací fire fire a zapomenutí klienta, kde je klient oddělen od dostupnosti služby a funkcí, jako např. přesměrování odpovědí a vícesměrovacích uzlů.
- Spolehlivé charakteristiky škálování: Systém zpráv založený na frontě umožňuje předvídatelné přidání kapacity do systému.
- Kvalita služby: Zprávy jsou hmatatelné a vysledovatelné a lze je snadno spravovat a spravovat.

Zprávy přenesené pomocí formátu zprávy Non-SOAP/Non-JMS (Pure MQMessage)

Pokud použijete vlastní kanál produktu IBM MQ pro prostředek WCF jako rozhraní jiného typu než SOAP pro přenos nativních zpráv produktu IBM MQ , budou zprávy přeneseny pomocí formátu zpráv Non-SOAP/ Non-JMS (Pure MQMessage) produktu IBM MQ.

Uživatelé služby WCF mohou službu spustit, nebo jinými slovy, uživatelé služby mohou odesílat zprávy do fronty produktu IBM MQ pomocí zpráv MQMessages. Aplikace mohou získat a nastavit pole MQMD a informační obsah. Je-li tato zpráva k dispozici ve frontách produktu IBM MQ MQ , může být tato zpráva zpracována všemi aplikacemi služby WCF nebo aplikacemi jiného typu než WCF, jako jsou aplikace typu C nebo Java spuštěné v produktu Windows, UNIX nebo z/OS.

Softwarové požadavky a pokyny k instalaci pro vlastní kanál IBM MQ pro WCF

Toto téma popisuje softwarové požadavky a informace o instalaci pro vlastní kanál IBM MQ pro WCF.

Vlastní kanál produktu IBM MQ pro službu WCF se může připojit pouze k produktu IBM WebSphere MQ 7.0 nebo k vyšším správcům front.

Softwarové požadavky na vlastní kanál WCF pro produkt IBM MQ

V této části jsou uvedeny softwarové požadavky pro vlastní kanál WCF pro produkt IBM MQ.

Běhové prostředí

- Produkt Microsoft.NET Framework v3.5 nebo vyšší musí být instalován na hostitelském počítači.
- *Java a .NET Messaging and Web Services* se standardně instaluje jako součást instalačního programu produktu IBM MQ 8.0 . Instaluje montážní celky produktu .NET potřebné pro vlastní kanál do mezipaměti Global Assembly Cache.

Poznámka: Pokud není před instalací produktu IBM MQ 8.0 nebo novější nainstalován produkt Microsoft.NET Framework v3.5 nebo vyšší, instalace produktu IBM MQ bude pokračovat bez chyby, ale vlastní kanál produktu IBM MQ je nedostupný. Je-li produkt .NET Framework nainstalován po instalaci produktu IBM MQ 8.0 nebo novější, musí být vlastní kanál produktu IBM MQ aktivován spuštěním skriptu *WMQInstallDir*\bin\amqiRegisterdotNet.cmd , kde *WMQInstallDir* je adresář, kde je nainstalován produkt IBM MQ 8.0 nebo novější. Tento skript nainstaluje vyžadované sestavení v mezipaměti GAC (Global Assembly Cache). Sada souborů amqi*.log zaznamenávající provedené akce se vytvoří v adresáři %TEMP% . It is not necessary to rerun the amqiRegisterdotNet.cmd script if .NET is upgraded to v3.5 or higher from an earlier version, for example, from .NET v2.0.

Vývojové prostředí

- Microsoft Visual Studio 2008 nebo Windows Software Development Kit for .NET 3.5 nebo novější.
- Na hostitelském počítači musí být nainstalován produkt Microsoft.NET Framework V3.5 nebo vyšší, aby bylo možné sestavit ukázkové soubory řešení.

Poznámka: Pokud není před instalací produktu IBM MQ 8.0 nebo novější nainstalován produkt Microsoft.NET Framework v3.5 nebo vyšší, instalace produktu IBM MQ bude pokračovat bez chyby, ale vlastní kanál produktu IBM MQ je nedostupný. Je-li produkt .NET Framework nainstalován po instalaci produktu IBM MQ 8.0 nebo novější, musí být vlastní kanál produktu IBM MQ aktivován spuštěním skriptu *WMQInstallDir*\bin\amqiRegisterdotNet.cmd , kde *WMQInstallDir* je adresář, kde je nainstalován produkt IBM MQ 8.0 nebo novější. Tento skript nainstaluje vyžadované sestavení v mezipaměti GAC (Global Assembly Cache). Sada souborů amqi*.log zaznamenávající provedené akce se vytvoří v adresáři %TEMP% . It is not necessary to rerun the amqiRegisterdotNet.cmd script if .NET is upgraded to v3.5 or higher from an earlier version, for example, from .NET v2.0.

Vlastní kanál produktu IBM MQ pro službu WCF: Co je nainstalováno?

Vlastní kanál pro produkt IBM MQ je transportním kanálem s jednotným programovacím modelem produktu Microsoft Windows Communication Foundation (WCF). Vlastní kanál je standardně instalován jako součást instalace produktu IBM MQ 8.0 nebo novější.

Vlastní kanál produktu IBM MQ pro prostředek WCF

Vlastní kanál produktu IBM MQ pro nástroj WCF je standardně instalován jako součást instalace produktu IBM MQ 8.0 . Vlastní kanál a jeho závislosti jsou obsaženy v komponentě Java and .NET Messaging and Web Services , která je standardně nainstalována. Při přechodu na verzi produktu IBM MQ 8.0 ze starší verze aktualizuje aktualizace standardně vlastní kanál produktu IBM MQ pro prostředek WCF, pokud byla komponenta produktu Java and .NET Messaging and Web Services již dříve nainstalována v dřívější instalaci.

Komponenta .NET Messaging and Web Services obsahuje soubor IBM.XMS.WCF.dll a soubor IBM.WMQ.WCF.dll a tyto soubory jsou hlavní vlastní montážní komplet kanálu, který obsahuje třídy rozhraní WCF. Tyto soubory jsou instalovány v mezipaměti GAC (Global Assembly Cache) a jsou také dostupné v následujícím adresáři: *MQ_INSTALLATION_PATH* \bin kde *MQ_INSTALLATION_PATH* je adresář, ve kterém je nainstalován produkt IBM MQ .

Následující tabulka shrnuje klíčové třídy, které jsou nezbytné pro použití vlastního kanálu.

Tabulka 177. Klíčové třídy vyžadované pro použití vlastního kanálu		
	Rozhraní SOAP/JMS (existující)	**NeSOAP/Non-JMS rozhraní (od IBM MQ 8.0)**
Vlastní sestavení kanálu	IBM.XMS.WCF.dll	IBM.WMQ.WCF.dll
Název vazby přenosu	IBM.XMS.WCF.SoapJmsIbmTransportBindingElement	IBM.WMQ.WCF.WmqIbmTransportBindingElement
Vazba importu vazeb	IBM.XMS.WCF.SoapJmsIbmTransportBindingElementImporter	IBM.WMQ.WCF.WmqIbmTransportBindingElementImporter
Konfigurace vazby přenosu	IBM.XMS.WCF.SoapJmsIbmTransportBindingElementConfig	IBM.WMQ.WCF.WmqIbmTransportBindingElementConfig
Ukázky (jednocestný)	SimpleOneWay_Client, SimpleOneWay_Service	Objekt MQMessaging_OneWay_Client, MQMessaging_OneWay_Service
Ukázky (RequestReply)	SimpleRequestReply_Client, SimpleRequestReplay_Service.	Objekt MQMessaging_RequestReply_Client, MQMessaging_RequestReply_Service

IBM.WMQ.WCF.dll podporuje rozhraní SOAP/JMS i Non-SOAP/Non-JMS . Vyspělé nové aplikace se doporučuje používat IBM.WMQ.WCF podporuje obě rozhraní.

Odesílání formátovaných zpráv MQSTR

V 9.0.5

From IBM MQ 9.0.5, if the request message is of type MQSTR, you can select to send the reply message in MQSTR format.

Chcete-li změnit formát zprávy odpovědi, musíte použít další parametr identifikátoru URI **replyMessageFormat** . Podporované hodnoty jsou:

""

"" je výchozí hodnota.

Zpráva odpovědi je ve formátu bajtu (MQMFT_NONE). Příklad:

```
"jms:/queue?
destination=SampleQ@QM1&connectionFactory=binding(server)connectQueueManager(QM1)
&initialContextFactory=com.ibm.mq.jms.Nojndi&replyDestination=SampleReplyQ&replyMessageForma
t= "
```

FUNKCE MQSTR

Zpráva odpovědi se nachází ve formátu MQSTR (MQMFT_STRING). Příklad:

```
"jms:/queue?
destination=SampleQ@QM1&connectionFactory=binding(server)connectQueueManager(QM1)
&initialContextFactory=com.ibm.mq.jms.Nojndi&replyDestination=SampleReplyQ&replyMessageForma
t=MQSTR"
```

Notes:

1. Hodnota pro **replyMessageFormat** nerozlišuje velká a malá písmena.
2. Použijete-li jinou hodnotu než "" nebo *MQSTR*, dojde k výjimce neplatné hodnoty parametru.

Ukázky vlastního kanálu produktu IBM MQ

Ukázky poskytují některé jednoduché příklady použití vlastního kanálu produktu IBM MQ pro prostředek WCF. Ukázky a jejich přidružené soubory jsou umístěny v adresáři *MQ_INSTALLATION_PATH*

`\tools\dotnet\samples\cs\wcf`, kde *MQ_INSTALLATION_PATH* je instalační adresář pro IBM MQ. Další informace o ukázkách vlastního kanálu produktu IBM MQ naleznete v tématu "Použití ukázek WCF" na stránce 1242.

svcutil.exe.config

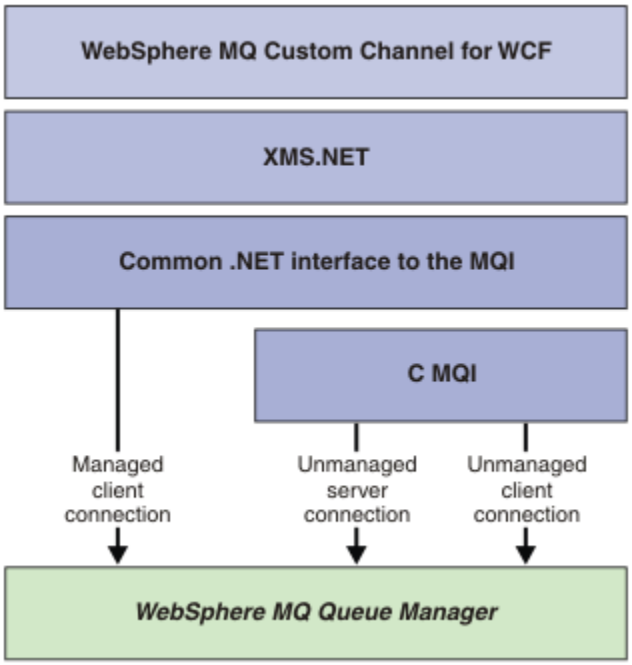
`svcutil.exe.config` je příkladem nastavení konfigurace vyžadovaného k povolení nástroje pro generování klienta proxy klienta Microsoft WCF `svcutil` k rozpoznání vlastního kanálu. Soubor `svcutil.exe.config` je umístěn v adresáři *MQ_INSTALLATION_PATH* `\tools\wcf\docs\examples\`, kde *MQ_INSTALLATION_PATH* je instalační adresář pro IBM MQ. Další informace o použití příkazu `svcutil.exe.config`naleznete v příručce "Generování serveru proxy klienta WCF a konfiguračních souborů aplikace pomocí nástroje svcutil s metadaty ze spuštěné služby" na stránce 1239.

Architektura WCF

Vlastní kanál produktu IBM MQ pro nástroj WCF je integrován v rozhraní API produktu `IBM Message Service Client for .NET (XMS .NET)`.

Rozhraní SOAP/JMS

Architektura WCF je zobrazena v následujícím diagramu:



Obrázek 161. Architektura WCF pro rozhraní SOAP/JMS

Pro produkt IBM WebSphere MQ 7.0.1 a novější jsou všechny požadované komponenty standardně instalovány spolu s instalací produktu.

Jsou to tři spojení:

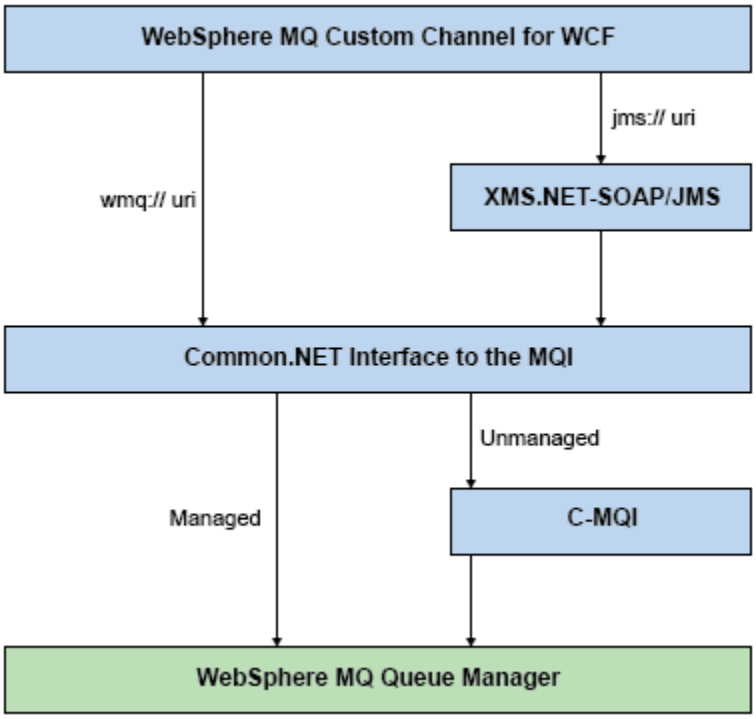
- Připojení spravovaného klienta
- Nespravovaná připojení k serveru
- Nespravovaná připojení klienta

Další informace o těchto spojeních najdete v tématu "Volby připojení WCF" na stránce 1229.

Rozhraní Non-SOAP/Non-JMS

Vlastní kanál produktu IBM MQ pro prostředek WCF podporuje rozhraní SOAP/JMS (dostupné z produktu IBM WebSphere MQ 7.0.1). a rozhraní Non-SOAP/Non-JMS .

Architektura WCF je zobrazena v následujícím diagramu:



Obrázek 162. Architektura WCF pro rozhraní Non-SOAP/Non-JMS

Použití vlastních kanálů produktu IBM MQ pro prostředek WCF

Přehled informací dostupných pro programátory používající IBM MQ vlastní kanály pro produkt Windows Communication Foundation (WCF).

Produkt Microsoft Windows Communication Foundation je základem pro webové služby a podporu systému zpráv v rámci Microsoft.NET Framework 3. Produkt IBM WebSphere MQ 7.0 nebo pozdější může být použit jako vlastní kanál v rámci WCF v prostředí .NET Framework 3 stejným způsobem jako vestavěné kanály nabízené produktem Microsoft.

Zprávy transportované přes vlastní kanál jsou formátovány podle implementace protokolu SOAP prostřednictvím produktu JMS produktu IBM WebSphere MQ 7.0 nebo novější. Aplikace pak mohou komunikovat se službami hostovanými službou WCF nebo infrastrukturou služby WebSphere SOAP přes JMS . Další informace o protokolu SOAP over JMSviz “Vyvíjení webových služeb s přenosem produktu IBM MQ pro SOAP” na stránce 1256.

Funkce a schopnosti vlastního kanálu služby WCF

V následujících tématech naleznete informace o funkcích a schopnostech vlastního kanálu služby WCF.

Vlastní tvary kanálů WCF

Přehled vlastních tvarů kanálů, které lze použít jako IBM MQ v rámci vlastních kanálů produktu Microsoft Windows Communication Foundation (WCF).

Vlastní kanál produktu IBM MQ pro službu WCF podporuje dva tvary kanálů:

- Jednosměrná
- Požadavek-odpověď

WCF automaticky vybere tvar kanálu podle hostované smlouvy služby.

Zakázky, které zahrnují metody, které používají pouze parametr **IsOneWay** , jsou obsluhovány jednosměrným tvarem kanálu, například:

```
[OperationContract(IsOneWay = true)]
void printString(String text);
```

Smlouvy, které zahrnují buď směs jednocestné a typu požadavek-odezva, nebo všechny metody požadavek-odpověď, jsou obsluhovány ve tvaru kanálu požadavek-odezva. Příklad:

```
    [OperationContract]
    int subtract(int a, int b);

    [OperationContract(IsOneWay = true)]
    void printString(string text);
```

Poznámka: Míšujete-li jednosměrné metody a metody požadavek-odezva ve stejné smlouvě, musíte zajistit, aby chování bylo zamýšleno, zejména při práci ve smíšeném prostředí, protože jednosměrné metody čekají, dokud neobdrží od služby odpověď s hodnotou null.

Jednosměrný kanál

Vlastní kanál IBM MQ jednosměrného kanálu pro WCF se používá například k odesílání zpráv z klienta WCF pomocí tvaru jednosměrného kanálu. Kanál může odesílat zprávy pouze jedním směrem, například ze správce front klienta do fronty ve službě WCF.

Kanál požadavků požadavku

Vlastní kanál požadavku IBM MQ pro WCF se používá například k asynchronnímu odesílání zpráv ve dvou směrech; pro asynchronní zasílání zpráv musí být použita stejná instance klienta. Kanál může odesílat zprávy jedním směrem, například od správce front klienta do fronty v rámci služby WCF a poté odeslat zprávu odpovědi z WCF do fronty ve správci front klienta.

Názvy a hodnoty parametrů identifikátoru URI WCF

Názvy a hodnoty parametrů identifikátoru URI pro rozhraní SOAP/JMS a rozhraní Non-SOAP/Non JMS .

Rozhraní SOAP/JMS

connectionFactory
: Parametr connectionFactory je povinný. Syntaxe tohoto parametru naleznete v tématu Syntaxe identifikátoru URI a parametry implementace webové služby.

InitialContextFactory
: Parametr továrny initialContextje povinný a musí být nastaven na hodnotu "com.ibm.mq.jms.Nojndi" kvůli kompatibilitě s produktem WebSphere Application Server a dalšími produkty (viz “Implementace služby do produktu WebSphere Application Server pro použití přenosu WebSphere pro protokol SOAP” na stránce 1308).

Rozhraní Non-SOAP/Non JMS

Formát identifikátoru URI je určen pro specifikace MA93 . Další podrobnosti o specifikacích produktu IBM MQ IRI viz SupportPac - MA93 .

Syntaxe identifikátoru URI IBM MQ

```
wmq-iri = "wmq:" [ "//" connection-name ] "/" wmq-dest ["?" parm *("&" parm)]
connection-name = tcp-connection-name / other-connection-name
tcp-connection-name = ihost [ ":" port ]
other-connection-name = 1*(iunreserved / pct-encoded)
wmq-dest = queue-dest / topic-dest
```

```
queue-dest = "msg/queue/" wmq-queue ["@" wmq-qmgr]
wmq-queue = wmq-name
wmq-qmgr = wmq-name
wmq-name = 1*48( wmq-char )
topic-dest = "msg/topic/" wmq-topic
wmq-topic = segment *( "/" segment )
```

IBM MQ IRI příklad

Následující příklad IRI sdělí žadateli služby, že může použít připojení vazby klienta TCP IBM MQ k počítači s názvem example.com na portu 1414 a vložit trvalé zprávy požadavků do fronty s názvem SampleQ ve správci front QM1. Funkce IRI určuje, že poskytovatel služby vloží odpovědi do fronty s názvem SampleReply.

```
1)wmq://example.com:1414/msg/queue/SampleQ@QM1?
ReplyTo=SampleReplyQ&persistence=MQPER_NOT_PERSISTENT
2)wmq://localhost:1414/msg/queue/Q1?
connectQueueManager=QM1&replyTo=Q2&connectionmode=managed
```

Pro připojení s povoleným protokolem TLS

Chcete-li zpřístupnit zabezpečená (TLS) připojení pomocí WCF Client/Service, nastavte následující vlastnosti s příslušnými hodnotami v identifikátoru URI. Všechny vlastnosti, které mají předponu "*", jsou povinné pro vytvoření zabezpečeného připojení.

- **sslKeyRepository**: *SYSTEM nebo *USER
- * **sslCipherSpec**: platnou CipherSpec, například TLS_RSA_WITH_AES_128_CBC_SHA256.
- **sslCertRevocationCheck**: true nebo false.
- **sslKeyResetCount**: hodnota větší než 32kb.
- **sslPeerName:** rozlišující název certifikátu serveru

Příklad:

```
"wmq://localhost:1414/msg/queue/SampleQ?
connectQueueManager=QM1&sslkeyrepository=*SYSTEM&sslcipherspec=
TLS_RSA_WITH_AES_128_CBC_SHA&sslcertrevocationcheck=true&"sslpe
ername=" + "" + "CN=ibmwebspheremqqmm&sslkeyresetcount=45000"
```

Doručení vlastního kanálu WCF

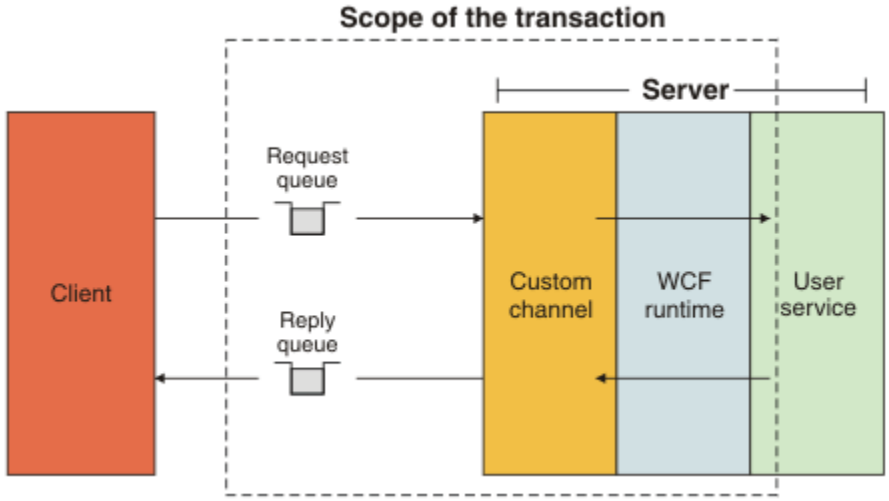
Zajištěné doručení zaručuje, že požadavek na službu nebo odpověď se bude provádět a nebude ztracena.

Je přijata zpráva požadavku a každá zpráva odpovědi se odešle pod bodem synchronizace lokální transakce, která může být odvolána v případě selhání běhového prostředí. Příklady těchto selhání jsou: Neošetřená výjimka vyvolaná službou, selhání při odeslání zprávy do služby nebo selhání doručení zprávy odpovědi.

`AssuredDelivery` je atribut zajištěného doručení, který lze uvést na servisní smlouvě, aby zaručil, že všechny zprávy vzniklé při zpracování požadavku přijaté službou a každá zpráva odpovědi odeslané ze služby se neztratí v případě selhání za běhu.

Aby se zajistilo, že zprávy budou zachovány i v případě selhání systému nebo výpadku proudu, musí být zprávy odeslány jako trvalé. Chcete-li používat trvalé zprávy, musí mít klientská aplikace tuto volbu určenou na svém identifikátoru URI koncového bodu. Další informace o nastavení vlastností identifikátoru URI viz: Syntaxe identifikátoru URI a parametry pro implementaci webové služby.

Distribuované transakce nejsou podporovány a rozsah transakce se nerozšiřuje nad zpracování požadavku a odpovědi prováděné produktem IBM MQ. Jakákoli práce provedená v rámci služby může být znovu spuštěna jako výsledek selhání, který způsobí, že zpráva bude přijata znovu. Následující diagram zobrazuje rozsah transakce:



Zajištěné doručení je povoleno použitím atributu `AssuredDelivery` pro třídu služeb, jak je zobrazeno v následujícím příkladu:

```
[AssuredDelivery]
class TestCalculatorService : IWMQSampleCalculatorContract
{
    public int add(int a, int b)
    {
        int ans = a + b;
        return ans;
    }
}
```

Používáte-li atribut `AssuredDelivery` , musíte mít na paměti následující body:

- Když kanál zjistí, že se selhání bude opakovat, pokud byla zpráva odvolána a přijata znovu, zpráva se bude považovat za nezpracovatelnou zprávu a nevrátí se do fronty požadavků na přepracování. Například: Pokud přijatá zpráva není správně formátovaná nebo nemůže být odeslána do služby. Neošetřené výjimky vyvolané z operace služby jsou vždy znovu odeslány, dokud nebyla zpráva znovu doručena do maximálního počtu, který je uvedený ve vlastnosti prahové hodnoty vrácení fronty požadavků. Další informace viz: “nezpracovatelné zprávy kanálu vlastního kanálu WCF” na stránce 1227
- Kanál provádí čtení, zpracování a odpovědi každé zprávy s požadavkem jako atomickou operaci pomocí jednoho podprocesu provádění k vynucení integrity transakcí. Chcete-li povolit spouštění operací služby souběžně, kanál umožňuje serveru WCF vytvořit více instancí kanálu. Počet instancí kanálu, které jsou k dispozici pro zpracování požadavků, je řízen vlastností vazby `MaxConcurrentCalls`. Další informace viz: “Volby konfigurace vazby WCF” na stránce 1235
- Funkce assured delivery používá jak IOperationInvoker , tak i IErrorHandler -body rozšiřitelnosti WCF. Pokud jsou tyto body rozšiřitelnosti používány externě aplikací, aplikace musí zajistit, aby byly volány všechny dříve registrované rozšiřitelné body. Pokud tak neučiníte pro IErrorHandler , může dojít k neohlášenému hlášení chyb. Pokud tak neučiníte, může produkt IOperationInvoker způsobit, že se WCF přestane reagovat.

Vlastní zabezpečení kanálu WCF

Vlastní kanál produktu IBM MQ pro službu WCF podporuje použití TLS pouze pro nespravovaná připojení klienta ke správci front.

TLS může být uvedeno jedním ze dvou způsobů:

- Uveďte TLS přímo na identifikátoru URI protokolu SOAP přes JMS . Úplný popis voleb TLS viz TLS a přenos IBM MQ pro SOAP .
- Uveďte TLS pomocí záznamu v tabulce definic kanálů klienta (CCDT). Další informace o CCDT viz Tabulka definic kanálů klienta

Tabulky definic kanálů klienta WCF (CCDT)

Vlastní kanál produktu IBM MQ pro službu WCF podporuje použití tabulek CCDT (Client Channel Definition CCDT) ke konfiguraci informací o připojení pro klientská připojení.

CCDTs jsou řízeny těmito dvěma proměnnými prostředí:

- *MQCHLLIB* určuje adresář, ve kterém je umístěna tabulka.
- *MQCHLTAB* určuje název souboru tabulky.

Tabulku definic kanálů nelze určit přímo v identifikátoru URI protokolu SOAP prostřednictvím produktu JMS . Jsou-li tyto proměnné prostředí definovány, pak mají přednost před všemi podrobnostmi o připojení klienta uvedenými v identifikátoru URI.

Další informace o tabulkách definic kanálů klienta naleznete v tématu: Tabulka definic kanálů klienta.

Související informace

Tabulka definic kanálů klienta

nezpracovatelné zprávy kanálu vlastního kanálu WCF

Pokud služba selže při zpracování zprávy požadavku nebo selže doručení zprávy odpovědi do fronty odpovědí, bude zpráva považována za nezpracovatelnou zprávu.

Nezpracovatelné zprávy požadavku

Pokud zprávu požadavku nelze zpracovat, bude s ní zacházeno jako s nezpracovatelnou zprávou. Tato akce zabrání tomu, aby služba znovu obdržela stejnou znovu zpracovatelnou zprávu. Pro nezpracovatelnou zprávu požadavku, která má být považována za nezpracovatelnou zprávu, musí být splněna jedna z následujících situací:

- Počet vrácení zpráv překročil prahovou hodnotu odvolání uvedenou ve frontě požadavků, která se vyskytne pouze, pokud byla pro službu uvedena zajištěná doručení. Další informace o zajištěného doručení najdete v tématu: “Doručení vlastního kanálu WCF” na stránce 1225
- Zpráva nebyla správně naformátována a nelze ji interpretovat jako zprávu protokolu SOAP over JMS .

Nezpracovatelné zprávy odpovědi

Pokud služba nedoručí zprávu odpovědi do fronty odpovědí, je zpráva odpovědi považována za nezpracovatelnou zprávu. U zpráv s odpovědí umožňuje tato akce později získat zprávy odpovědí za účelem určení příčiny pomoci.

Zpracování nezpracovatelných zpráv

Akce provedená pro nezpracovatelnou zprávu závisí na konfiguraci správce front a na hodnotách nastavených ve volbách sestavy ve zprávě. Pro SOAP přes JMSjsou standardně nastaveny následující volby sestavy na základě zpráv požadavku a nejsou konfigurovatelné:

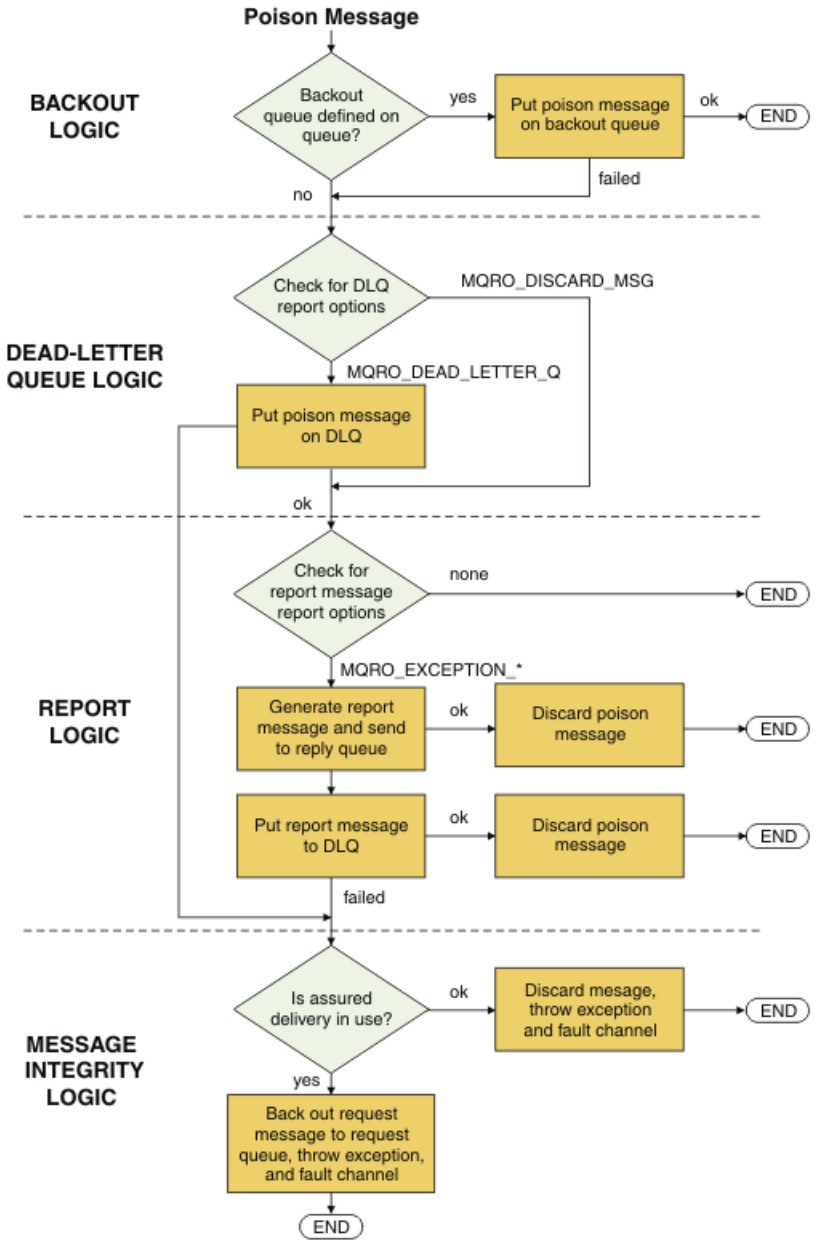
- MQRO_EXCEPTION_WITH_FULL_DATA
- MQRO_EXPIRATION_WITH_FULL_DATA
- MQRO_DISCARD_MSG

Pro SOAP přes JMSse standardně nastavuje následující volba sestavy ve zprávách odpovědí a není konfigurovatelná:

- MQRO_DEAD_LETTER_Q

Pokud zprávy pocházejí ze zdroje mimo WCF, odkazujte se na dokumentaci pro tento zdroj.

Následující diagram zobrazuje možné akce a kroky, které podnikli v případě selhání zpracování nezpracovatelných zpráv:



Schopnosti zpráv produktu IBM MQ pro aplikace WCF

Non-SOAP/Non-JMS (to znamená, IBM MQ) schopnosti zpráv pro aplikace WCF.

V případě rozhraní Non-SOAP/Non-JMS jsou funkce zpráv produktu IBM MQ pro aplikace WCF následující:

- Aplikace WCF mohou odesílat a přijímat základní zprávy produktu IBM MQ , které mohou být zpracovány libovolnou aplikací produktu IBM MQ .
- Aplikace WCF mají plnou kontrolu k aktualizaci MQMD a payload.
- The WCF client can send IBM MQ messages that can be consumed by any IBM MQ clients, for example C, Java, JMS, and .NET clients.

Rozhraní WCF pro jiné rozhraní než SOAP/Non-JMS musí používat následující třídy pro nastavení informačního obsahu zprávy a MQMD pro zprávu:

- WmqStringZpráva pro informační obsah typu String.
- WmqBytesZpráva pro informační obsah typu Bytes
- WmqXmlZpráva pro informační obsah typu XML

Chcete-li nastavit informační obsah zprávy, použijte vlastnost **Data** pro zprávu WmqString, WmqBytesMessage nebo WmqXmlMessage třídy, v závislosti na typu informačního obsahu. Chcete-li například nastavit informační obsah typu String, použijte následující kód:

```
WmqStringMessage strMsg = new WmqStringMessage();
//Setting the Message PayLoad
strMsg.Data = "Hello World";
//MQMD property
strMsg.Format = WmqMessageFormat.MQFMT_STRING;
```

Volby připojení WCF

K dispozici jsou tři režimy připojení vlastního kanálu produktu IBM MQ pro službu WCF ke správci front. Zvažte, který typ připojení nejlépe vyhovuje vašim požadavkům.

Další informace o volbách připojení najdete v tématu: “Rozdíly spojení” na stránce 529

Další informace o architektuře WCF naleznete v tématu: “Architektura WCF” na stránce 1222

Nespravované připojení klienta

Připojení vytvořené v tomto režimu se připojuje jako klient IBM MQ k serveru IBM MQ spuštěnému buď na lokálním počítači, nebo na vzdáleném počítači.

Chcete-li použít vlastní kanál IBM MQ pro WCF jako klienta IBM MQ , můžete jej instalovat se serverem IBM MQ MQI clientbuď na server IBM MQ , nebo na samostatném počítači.

Nespravované připojení k serveru

Při použití v režimu vázání serveru používá vlastní kanál produktu IBM MQ pro službu WCF rozhraní API správce front, a nikoli komunikace prostřednictvím sítě. Použití připojení vazeb poskytuje lepší výkon pro aplikace produktu IBM MQ než použití síťových připojení.

Chcete-li používat připojení vazeb, je třeba nainstalovat vlastní kanál produktu IBM MQ pro prostředek WCF na serveru IBM MQ .

Připojení spravovaného klienta

Připojení vytvořené v tomto režimu se připojuje jako klient IBM MQ k serveru IBM MQ spuštěnému buď na lokálním počítači, nebo na vzdáleném počítači.

Vlastní třídy kanálu produktu IBM MQ pro produkt .NET 3 připojující se v tomto režimu zůstávají ve spravovaném kódu produktu .NET a nevytvářejí žádná volání pro nativní služby. Další informace o spravovaném kódu naleznete v dokumentaci Microsoft .

Pro použití spravovaného klienta existuje několik omezení. Další informace o těchto omezeních viz “Připojení spravovaného klienta” na stránce 529.

Vytvoření a konfigurace vlastního kanálu produktu IBM MQ pro prostředek WCF

Vlastní kanály produktu IBM MQ pro službu WCF pracují stejným způsobem jako kanály služby WCF nabízené produktem Microsoft. Vlastní kanál produktu IBM MQ pro službu WCF lze vytvořit jedním ze dvou způsobů.

Informace o této úloze

Vlastní kanál produktu IBM MQ se integruje s WCF jako přenosovým kanálem WCF a jako takový musí být spárován s kódovacím programem zpráv a volitelnými kanály protokolu, takže může vytvořit kompletní zásobník kanálu, který může aplikace použít. Pro úspěšné vytvoření úplné sady kanálů jsou vyžadovány dva prvky:

1. Definice vázání: Určuje, které prvky jsou vyžadovány pro sestavení sady kanálů aplikací, včetně transportního kanálu, kódování zpráv a jakýchkoli protokolů a veškerých obecných nastavení konfigurace. Pro vlastní kanál musí být definice vazby vytvořena ve formě přizpůsobené vazby WCF.
2. Definice koncového bodu: Odkazuje na smlouvu na službu s definicí vazby a také poskytuje skutečný identifikátor URI připojení, který popisuje, kde se aplikace může připojit. Pro vlastní kanál je identifikátor URI ve formě identifikátoru URI protokolu SOAP prostřednictvím produktu JMS .

Tyto definice lze vytvořit jedním ze dvou způsobů:

- Administrativně; Definice se vytvářejí poskytnutím podrobností v konfiguračním souboru aplikace (například: `app.config`).
- Programové; Definice jsou vytvářeny přímo z kódu aplikace.

Rozhodnutí o tom, jaká metoda použít k vytvoření definic musí být založena na požadavcích na aplikaci:

- Administrativní metoda konfigurace poskytuje flexibilitu pro změnu podrobností o službách a klientu po implementaci bez opětovného sestavení aplikace.
- Programová metoda pro konfiguraci poskytuje větší ochranu před chybami konfigurace a schopnost dynamicky generovat konfiguraci za běhu.

Vytvoření vlastního uživatelského kanálu WCF administrativně dodáním informací o vazbách a koncových bodech v konfiguračním souboru aplikace

Vlastní kanál produktu IBM MQ pro službu WCF je kanál WCF na úrovni transportu. Aby bylo možné použít vlastní kanál, musí být definován koncový bod a vazba a tyto definice lze provést dodáním informací o vazbách a koncových bodech v konfiguračním souboru aplikace.

Chcete-li konfigurovat a používat vlastní kanál produktu IBM MQ pro službu WCF, což je kanál WCF na úrovni transportu, musí být definována vazba a definice koncového bodu. Vazba obsahuje informace o konfiguraci kanálu a definice koncového bodu obsahuje podrobnosti o připojení. Tyto definice lze vytvořit dvěma způsoby:

- Programově přímo z kódu aplikace, jak je popsáno zde: “Vytvoření vlastního kanálu WCF nahrazením informací o vazbách a koncovém bodu programově” na stránce 1232
- Administrativně tak, že poskytnete podrobnosti v konfiguračním souboru aplikace, jak je popsáno v následujícím postupu.

Konfigurační soubor aplikace klienta nebo služby se běžně nazývá *yourappname*`.exe.config` , kde *yourappname* je název vaší aplikace. Konfigurační soubor aplikace je nejsnáze modifikován s použitím nástroje editoru konfigurace služby Microsoft nazvaného `SvcConfigEditor.exe` následujícím způsobem:

- Spusťte nástroj editoru konfigurace produktu `SvcConfigEditor.exe` . Výchozí umístění instalace pro nástroj je: *`Drive:`*`\Program Files\Microsoft SDKs\Windows\v6.0\Bin\SvcConfigEditor.exe` , kde *Drive:* je název instalační jednotky.

Krok 1: Přidání rozšíření prvku vazby k povolení prostředku WCF k vyhledání vlastního kanálu

1. Klepnutím na volbu **Rozšířené** > **Rozšíření** > **prvek vazby** otevřete nabídku a vyberte volbu **Nový** .
2. Vyplňte pole, jak je uvedeno v této tabulce:

Tabulka 178. Nová pole prvku vazby

Pole	**Hodnota**
Název	`IBM.XMS.WCF.SoapJmsIbmTransportChannel`

Tabulka 178. Nová pole prvku vazby (pokračování)	
Pole	**Hodnota**
Typ	Přejděte do produktu `IBM.XMS.WCF.dll` v mezipaměti GAC (Global Assembly Cache) a vyberte volbu `IBM.XMS.WCFSoapJmsIbmTransportBindingElementConfig`.

Krok 2: Vytvoření vlastní definice vázání, která páruje vlastní kanál s kódováním zpráv WCF

1. Klepnutím pravým tlačítkem myši na volbu **Vazby** otevřete nabídku a vyberte volbu **Nová konfigurace vazby**.
2. Vyplňte pole, jak je uvedeno v této tabulce:

Tabulka 179. Nová pole konfigurace vazby	
Pole	**Hodnota**
Název	`CustomBinding_WMQ`
BindingElement 1	`textMessageEncoding (MessageVersion: Soap11)`
BindingElement 2	`IBM.XMS.WCF.SoapJmsIbmTransportChannel`

Krok 3: Určete vlastnosti vázání

1. Vyberte *IBM.XMS.WCF.SoapJmsIbmTransportChannel* vazba přenosu z vazby, kterou jste vytvořili v: “Krok 2: Vytvoření vlastní definice vázání, která páruje vlastní kanál s kódováním zpráv WCF” na stránce 1231
2. Proveďte všechny požadované změny výchozích hodnot vlastností, jak je popsáno v: “Volby konfigurace vazby WCF” na stránce 1235

Krok 4: Vytvoření definice koncového bodu

Vytvořte definici koncového bodu, která odkazuje na vlastní vazbu, kterou jste vytvořili v: “Krok 2: Vytvoření vlastní definice vázání, která páruje vlastní kanál s kódováním zpráv WCF” na stránce 1231, a poskytuje podrobnosti o připojení služby. Způsob, jakým jsou tyto informace zadány, závisí na tom, zda je definice pro klientskou aplikaci nebo pro aplikaci služeb.

Pro klientskou aplikaci přidejte definici koncového bodu do sekce klienta následujícím způsobem:

1. Klepněte pravým tlačítkem myši na nabídku **Klient** > **Koncové body**, abyste otevřeli nabídku a vyberte volbu **Nový koncový bod klienta**.
2. Vyplňte pole, jak je uvedeno v této tabulce:

Tabulka 180. Pole nového koncového bodu klienta	
Pole	**Hodnota**
Název	`Endpoint_WMQ`
Adresa	*Identifikátor URI SOAP/JMS popisující podrobnosti připojení WMQ vyžadované pro přístup ke službě.* Další podrobnosti viz: “Uživatelský kanál IBM MQ pro formát adresy identifikátoru URI koncového bodu služby WCF” na stránce 1234
Vazba	`customBinding`

Tabulka 180. Pole nového koncového bodu klienta (pokračování)	
Pole	**Hodnota**
BindingConfiguration	`CustomBinding_WMQ`
Smlouva.	*Název rozhraní smlouvy služeb*

Pro aplikaci služby přidejte do sekce služeb definici služby takto:

1. Klepnutím na tlačítko **Služby** otevřete nabídku a vyberte volbu **Nová služba**a poté vyberte třídu služeb, kterou chcete hostovat.
2. Přidejte definici koncového bodu do sekce **Koncové body** pro novou službu a doplňte pole tak, jak je uvedeno v této tabulce:

Tabulka 181. Nová pole koncového bodu služby	
Pole	**Hodnota**
Název	`Endpoint_WMQ`
Adresa	*Identifikátor URI SOAP/JMS popisující podrobnosti připojení WMQ vyžadované pro přístup ke službě.* Další podrobnosti viz: "Uživatelský kanál IBM MQ pro formát adresy identifikátoru URI koncového bodu služby WCF" na stránce 1234
Vazba	`customBinding`
BindingConfiguration	`CustomBinding_WMQ`
Smlouva.	*Název implementační třídy služby*

Vytvoření vlastního kanálu WCF nahrazením informací o vazbách a koncovém bodu programově

Vlastní kanál produktu IBM MQ pro službu WCF je kanál WCF na úrovni transportu. Koncový bod a vazba musí být definovány pro použití vlastního kanálu a tyto definice lze provádět programově přímo z kódu aplikace.

Chcete-li konfigurovat a používat vlastní kanál produktu IBM MQ pro službu WCF, což je kanál WCF na úrovni transportu, musí být definována vazba a definice koncového bodu. Vazba obsahuje informace o konfiguraci kanálu a definice koncového bodu obsahuje podrobnosti o připojení. Další informace viz "Použití ukázek WCF" na stránce 1242.

Tyto definice lze vytvořit dvěma způsoby:

- Administrativně tak, že poskytnete podrobnosti v konfiguračním souboru aplikace, jak je popsáno v tématu "Vytvoření vlastního uživatelského kanálu WCF administrativně dodáním informací o vazbách a koncových bodech v konfiguračním souboru aplikace" na stránce 1230.
- Programově přímo z kódu aplikace, jak je popsáno v následujících dílčích tématech.

Programově definování informací o vazbách a koncových bodech: Rozhraní SOAP/JMS

V případě rozhraní SOAP/JMS můžete definovat koncový bod a vazbu programově přímo z kódu aplikace.

Informace o této úloze

Chcete-li dodat informace o vazbě a koncovém bodu programově, přidejte do své aplikace požadovaný kód provedením následujících kroků.

Postup

1. Vytvořte instanci prvku vazby přenosu daného kanálu přidáním následujícího kódu do vaší aplikace:

```
SoapJmsIbmTransportBindingElement transportBindingElement = new
SoapJmsIbmTransportBindingElement();
```

2. Nastavte všechny vyžadované vlastnosti vazby, například přidáním následujícího kódu do vaší aplikace pro nastavení produktu `ClientConnectionMode`:

```
transportBindingElement.ClientConnectionMode = XmsWCFBindingProperty.AS_URI;
```

3. Vytvořte vlastní vazbu, která bude obsahovat dvojici transportního kanálu s kódováním zpráv přidáním následujícího kódu do vaší aplikace:

```
Binding binding = new CustomBinding(new TextMessageEncodingBindingElement(),
transportBindingElement);
```

4. Vytvořte identifikátor URI SOAP/JMS .

   Identifikátor URI SOAP/JMS , který popisuje podrobnosti připojení produktu IBM MQ požadované pro přístup ke službě, musí být uveden jako adresa koncového bodu. Zadaná adresa závisí na tom, zda se kanál používá pro aplikaci služby nebo pro klientskou aplikaci.

   - For client applications, the SOAP/JMS URI must be created as an `EndpointAddress` as follows:

     ```
     EndpointAddress address = new EndpointAddress("jms:/queue?
     destination=SampleQ@QM1&connectionFactory
     =connectQueueManager(QM1)&initialContextFactory=com.ibm.mq.jms.Nojndi");
     ```

   - Pro servisní aplikace musí být identifikátor URI SOAP/JMS vytvořen následujícím způsobem:

     ```
     Uri address = new Uri("jms:/queue?destination=SampleQ@QM1&connectionFactory=
     connectQueueManager(QM1)&initialContextFactory=com.ibm.mq.jms.Nojndi");
     ```

     Další informace o adresách koncových bodů viz “Uživatelský kanál IBM MQ pro formát adresy identifikátoru URI koncového bodu služby WCF” na stránce 1234.

Programové definování informací o vazbách a koncových bodech: rozhraní Non-SOAP/Non-JMS
V případě rozhraní Non-SOAP/Non-JMS můžete definovat koncový bod a vazbu programově přímo z kódu aplikace.

Informace o této úloze

Chcete-li dodat informace o vazbě a koncovém bodu programově, přidejte do své aplikace požadovaný kód provedením následujících kroků.

Postup

1. Vytvořte objekt WmqBinding přidáním následujícího kódu do vaší aplikace:

```
WmqBinding binding = new WmqBinding();
```

   Tento kód vytvoří vazbu, která páruje prvek WmqMsgEncodingElement a WmqIbmTransportBindingvyžadovaný pro rozhraní Non-SOAP/Non-JMS .

2. Zadejte identifikátor URI wmq: //, který popisuje podrobnosti připojení produktu IBM MQ potřebné pro přístup ke službě.

   Způsob zadání identifikátoru URI wmq: // závisí na tom, zda je kanál používán pro aplikaci služby nebo pro klientskou aplikaci.

- Pro klientské aplikace musí být identifikátor URI wmq: // vytvořen jako EndpointAddress takto:

```
EndpointAddress address = new EndpointAddress
("wmq://localhost:1414/msg/queue/Q1?connectQueueManager=QM1&replyTo=Q2");
```

- Pro servisní aplikace je třeba identifikátor URI wmq: // vytvořit jako identifikátor URI takto:

```
Uri sampleAddress = new Uri(
"wmq://localhost:1414/msg/queue/Q1?connectQueueManager=QM1&replyTo=Q2");
```

Uživatelský kanál IBM MQ pro formát adresy identifikátoru URI koncového bodu služby WCF

Webová služba je uvedena pomocí identifikátoru URI (Universal Resource Identifier), který poskytuje podrobnosti o umístění a připojení. Formát identifikátoru URI závisí na tom, zda používáte rozhraní SOAP/JMS nebo rozhraní Non-SOAP/Non-JMS .

Rozhraní SOAP/JMS

Formát identifikátoru URI, který je podporován v transportu produktu IBM MQ pro protokol SOAP, povoluje během přístupu k cílovým službám komplexní stupeň řízení parametrů a parametrů specifických pro typ SOAP/ IBM MQ . Tento formát je kompatibilní s WebSphere Application Server a s CICSusnadněním integrace produktu IBM MQ s oběma těmito produkty.

Syntaxe identifikátoru URI je následující:

```
jms:/queue? name=value&name=value...
```

, kde `název` je název parametru a *hodnota* je odpovídající hodnota, a prvek `name` = *value* může být opakován s druhým a následným výskytem, které předchází ampersand (&).

Další informace o nastavení vlastností identifikátorů URI naleznete v tématu Syntaxe identifikátoru URI a parametry pro implementaci webové služby .

Názvy parametrů rozlišují velikost písmen, stejně jako názvy objektů IBM MQ . Je-li některý parametr zadán vícekrát než jednou, bude konečným výskytem tohoto parametru účinek, což znamená, že klientské aplikace mohou hodnoty parametrů přepsat připojením k identifikátoru URI. Jsou-li zahrnuty jakékoli další nerozpoznané parametry, budou ignorovány.

Pokud ukládáte identifikátor URI do řetězce XML, musíte znázornit znak ampersand ve tvaru "&". Podobně platí, že pokud je identifikátor URI kódován ve skriptu, postarejte se o řídicí znaky, jako je **&** , které by jinak shell interpretoval.

Toto je příklad jednoduchého identifikátoru URI pro službu Axis:

```
jms:/queue?destination=myQ&connectionFactory=()
&initialContextFactory=com.ibm.mq.jms.Nojndi
```

Zde je příklad jednoduchého identifikátoru URI pro službu .NET :

```
jms:/queue?destination=myQ&connectionFactory=()&targetService=MyService.asmx
&initialContextFactory=com.ibm.mq.jms.Nojndi
```

Dodávané jsou pouze požadované parametry (`targetService` je povinný pouze pro služby produktu .NET) a `connectionFactory` nemá žádné volby.

V tomto příkladu Axis obsahuje `connectionFactory` řadu voleb:

```
jms:/queue?destination=myQ@myRQM&connectionFactory=connectQueueManager(myconnQM)
```

```
binding(client)clientChannel(myChannel)clientConnection(myConnection)
&initialContextFactory=com.ibm.mq.jms.Nojndi
```

V tomto příkladu Axis byla také zadána volba `sslPeerName` volby `connectionFactory` . Hodnota samotného názvu sslPeerobsahuje dvojice název-hodnota a významné vložené mezery:

```
jms:/queue?destination=myQ@myRQM&connectionFactory=connectQueueManager(myconnQM)
binding(client)clientChannel(myChannel)clientConnection(myConnection)
sslPeerName(CN=MQ Test 1,O=IBM,S=Hampshire,C=GB)
&initialContextFactory=com.ibm.mq.jms.Nojndi
```

Rozhraní NON-SOAP/Non-JMS

Formát identifikátoru URI pro rozhraní NON-SOAP/Non-JMS umožňuje komplexní stupeň řízení parametrů a voleb specifických pro produkt IBM MQ při přístupu k cílovým službám.

Syntaxe identifikátoru URI je následující:

```
wmq://example.com:1415/msg/queue/INS.QUOTE.REQUEST@MOTOR.INS ?ReplyTo=msg/queue/
INS.QUOTE.REPLY@BRANCH452&persistence=MQPER_NOT_PERSISTENT
```

Tato adresa IRI sděluje žadateli služby, že může použít připojení vazby klienta TCP IBM MQ k počítači s názvem example.com na portu 1415 a vložit trvalé zprávy požadavků do fronty s názvem INS.QUOTE.REQUEST ve správci front MOTOR.INS. Funkce IRI určuje, že poskytovatel služby vloží odpovědi do fronty s názvem INS.QUOTE.REPLY ve správci front BRANCH452. Formát identifikátoru URI je určen pro balík SupportPac MA93. Další podrobnosti o specifikacích IBM MQ IRI viz SupportPac MA93: IBM MQ -Definice služby .

Volby konfigurace vazby WCF

K dispozici jsou dva způsoby použití voleb konfigurace pro informace o vazbě vlastního kanálu. Buď nastavte vlastnosti administrativně, nebo je můžete nastavit programově.

Volby konfigurace vazby lze nastavit jedním ze dvou způsobů:

1. Administrativně: Nastavení vlastností vazby musí být určena v části transportu definice vlastní vazby v konfiguračním souboru aplikací, například: `app.config`.
2. Programově: Kód aplikace musí být upraven tak, aby určoval vlastnost při inicializaci vlastní vazby.

Administrativně nastavení vlastností vázání

Nastavení vlastností vazby lze zadat v konfiguračním souboru aplikace, například: `app.config`. Konfigurační soubor je generován produktem **`svcutil`**, jak je uvedeno v následujících příkladech.

Rozhraní SOAP/JMS

```
<customBinding>
...
      <IBM.XMS.WCF.SoapJmsIbmTransportChannel maxBufferPoolSize="524288"
      maxMessageSize="4000000" clientConnectionMode="0" maxConcurrentCalls="16"/>
...
</customBinding>
```

Rozhraní Non-SOAP/Non-JMS

```
<customBinding>
      <IBM.WMQ.WCF.WmqMsgEncodingElement/>
      <IBM.WMQ.WCF.WmqIbmTransportChannel maxBufferPoolSize="524288"
      maxMessageSize="65536" clientConnectionMode="managedclient"/>
</customBinding>
```

Programové nastavení vlastností vazby

Chcete-li přidat vlastnost vazby WCF k určení režimu připojení klienta, je třeba upravit kód služby tak, aby určoval vlastnost během inicializace vlastní vazby.

Při zadávání režimu připojení nespravovaného klienta použijte následující příklad:

```
SoapJmsIbmTransportBindingElement
transportBindingElement = new SoapJmsIbmTransportBindingElement();
transportBindingElement.ClientConnectionMode = XmsWCFBindingProperty.CLIENT_UNMANAGED;

Binding sampleBinding = new CustomBinding(new TextMessageEncodingBindingElement(),
                                                    transportBindingElement);
```

Vlastnosti vazby WCF

Tabulka 182. Hodnoty vlastností vazeb při nastavení administrativně nebo programově

Název vlastnosti	Aplikace klienta nebo služby	Administrativní hodnota	Programová hodnota	Popis
maxBufferPoolSize	Oboje	0 až 64 bitů podepsané celé číslo	0 až 64 bitů podepsané celé číslo	Určuje maximální velikost paměti, kterou lze použít k uložení vyrovnávacích pamětí zpráv WCF pro instanci kanálu.
Velikost maxMessage	Oboje	1 až 32 bitů signed integer	1 až 32 bitů signed integer	Určuje maximální velikost paměti, kterou lze použít pro jednotlivé zprávy WCF.
Režim clientConnection	Oboje	0 (Výchozí hodnota) 1	AS_URI (Výchozí hodnota) NESPRAVOVÁNO KLIENTA	Určuje režim připojení klienta transportního kanálu. Hodnota 0 znamená, že režim připojení klienta je stejný jako v identifikátoru URI. Použije se pouze v případě, že je použito připojení klienta. Určuje, že režim připojení klienta je určen v identifikátoru URI. Hodnota 0 je výchozí hodnotou, pokud není nastaven režim připojení klienta. Hodnota 1 znamená, že režim připojení klienta je nespravovaný klient. Použije se pouze v případě, že je použito připojení klienta.

Tabulka 182. Hodnoty vlastností vazeb při nastavení administrativně nebo programově (pokračování)

Název vlastnosti	Aplikace klienta nebo služby	Administrativní hodnota	Programová hodnota	Popis
Volání MaxConcurrent	Klient	Rozsah je 0-2 147 483 647 16 je výchozí hodnota	Rozsah je 0-2 147 483 647 16 je výchozí hodnota	Tato vlastnost definuje maximální počet souběžných operací, které mohou být prováděny na individuálním serveru proxy klienta najednou. Je-li spuštěno více operací, jsou řazeny do fronty, dokud nebude dokončena nebo ukončena operace probíhající nebo probíhající. Toto nastavení lze použít k řízení maximálního počtu podprocesů a prostředků, které může spotřebovávat jednotlivý server proxy. Hodnota 0 tento limit odebere, a umožní tak pokus o provedení všech operací současně.
Volání MaxConcurrent	Služba	Rozsah je 1-2 147 483 647 16 je výchozí hodnota	Rozsah je 1-2 147 483 647 16 je výchozí hodnota	Tato vlastnost se používá pouze v případě, že je povolena funkce zajištěného doručení (Další informace o zajištěného doručení viz “Doručení vlastního kanálu WCF” na stránce 1225). Určuje maximální počet souběžných operací, které mohou ve stejnou dobu pro daný koncový bod probíhat současně. Při změně tohoto nastavení je třeba věnovat pozornost. Každá souběžná operace vyžaduje další prostředky, zejména novou instanci vlastního kanálu a přidružené podprocesy z fondu podprocesů, aby bylo možné provést příslušné akce. Nadalokace může být kontraproduktivní a nepříznivě ovlivnit výkonnost. Aby bylo možné tuto vlastnost podporovat, musí být vytvořena odpovídající konfigurace fondu podprocesů.

Budování a hosting služeb pro WCF

Přehled služeb WCF (Microsoft Windows Communication Foundation) vysvětluje, jak vytvořit a nakonfigurovat služby WCF.

Vlastní kanál produktu IBM MQ pro službu WCF a služby WCF, které jej používají, může být hostován následujícími způsoby:

- Samohostitelství
- Služba Windows

Vlastní kanál produktu IBM MQ pro službu WCF nemůže být hostován ve službě Windows Process Activation Service.

Následující témata poskytují některé jednoduché příklady samohostování k demonstraci příslušných kroků. Online dokumentace produktu Microsoft WCF, která obsahuje další informace a nejnovější podrobnosti, naleznete na webu Microsoft MSDN na adrese https://msdn.microsoft.com.

Sestavení aplikací služeb WCF pomocí metody 1: vlastní hostování administrativně pomocí konfiguračního souboru aplikace

Poté, co jste vytvořili konfigurační soubor aplikace, otevřete instanci služby a přidejte do aplikace uvedený kód.

Než začnete

Vytvořte nebo upravte konfigurační soubor aplikace pro službu, jak je popsáno v: "Vytvoření vlastního uživatelského kanálu WCF administrativně dodáním informací o vazbách a koncových bodech v konfiguračním souboru aplikace" na stránce 1230

Informace o této úloze

1. Vytvořit instanci a otevřít instanci služby v hostiteli služby. Typ služby musí být stejný jako typ služby uvedený v konfiguračním souboru služby.
2. Přidejte do své aplikace tento kód:

```
ServiceHost service = new ServiceHost(typeof(MyService));
service.Open();
...
service.Close();
```

Sestavování aplikací služeb WCF pomocí metody 2: Samoobsluha programově přímo z aplikace

Přidejte vlastnosti vazby, vytvořte hostitele služby s instancí požadované třídy služeb a otevřete danou službu.

Než začnete

1. Přidejte odkaz na soubor vlastního kanálu `IBM.XMS.WCF.dll` do projektu. `IBM.XMS.WCF.dll` je v *WMQInstallDir*`\bin` , kde *WMQInstallDir* je adresář, ve kterém je nainstalován produkt IBM MQ .
2. Přidejte příkaz *using* do oboru názvů `IBM.XMS.WCF` , například: `using IBM.XMS.WCF`
3. Vytvořte instanci prvku vazby kanálů a koncového bodu, jak je popsáno v: "Vytvoření vlastního kanálu WCF nahrazením informací o vazbách a koncovém bodu programově" na stránce 1232

Informace o této úloze

Jsou-li vyžadovány změny vlastností vazby kanálu, postupujte takto:

1. Přidejte vlastnosti vazby do produktu `transportBindingElement` , jak ukazuje následující příklad:

```
SoapJmsIbmTransportBindingElement transportBindingElement = new
SoapJmsIbmTransportBindingElement();
Binding binding = new CustomBinding(new TextMessageEncodingBindingElement(),
transportBindingElement);
Uri address = new Uri("jms:/queue?destination=SampleQ@QM1&connectionFactory=

connectQueueManager(QM1)&initialContextFactory=com.ibm.mq.jms.Nojndi");
```

2. Vytvořte hostitele služby s instancí požadované třídy služeb:

```
ServiceHost service = new ServiceHost(typeof(MyService));
```

3. Otevřete službu:

```
service.AddServiceEndpoint(typeof(IMyServiceContract), binding, address);
service.Open();
...
service.Close();
```

Vystavení metadat pomocí koncového bodu HTTP

Pokyny pro vystavení metadat služby, která je nakonfigurována pro použití vlastního kanálu produktu IBM MQ pro službu WCF.

Informace o této úloze

Pokud metadata služeb musí být odkryta (aby mohly být nástroje, jako například produkt `svcutil` , k němu přistupovat přímo ze spuštěné služby spíše než ze souboru WSDL offline), musí být provedeny vystavením metadat služeb koncovým bodem protokolu HTTP. Následující kroky lze použít k přidání tohoto dalšího koncového bodu.

1. Přidejte základní adresu, kde musí být metadata vystavena objektu ServiceHost, například:

```
ServiceHost service = new ServiceHost(typeof(TestService),
                      new Uri("http://localhost:8000/MyService"));
```

2. Před otevřením služby přidejte následující kód do ServiceHost :

```
ServiceMetadataBehavior metadataBehavior = new ServiceMetadataBehavior();
  metadataBehavior.HttpGetEnabled = true;
  service.Description.Behaviors.Add(metadataBehavior);
  service.AddServiceEndpoint(typeof(IMetadataExchange),
     MetadataExchangeBindings.CreateMexHttpBinding(), "mex");
```

Výsledky

Metadata jsou nyní k dispozici na této adrese: `http://localhost:8000/MyService`

Sestavování aplikací klienta pro WCF

Přehled generování a sestavení klientských aplikací produktu Microsoft Windows Communication Foundation (WCF).

Klientská aplikace může být vytvořena pro službu WCF; klientské aplikace se obvykle generují pomocí obslužného nástroje Microsoft ServiceModel Metadata Utility Tool (Svcutil.exe) k vytvoření požadované konfigurace a souborů serveru proxy, které mohou být použity přímo aplikací.

Generování serveru proxy klienta WCF a konfiguračních souborů aplikace pomocí nástroje svcutil s metadaty ze spuštěné služby

Pokyny pro použití nástroje Microsoft svcutil.exe ke generování klienta pro službu, která je nakonfigurována pro použití vlastního kanálu produktu IBM MQ pro službu WCF.

Než začnete

K dispozici jsou tři předpoklady pro použití nástroje svcutil k vytvoření požadované konfigurace a souborů proxy, které lze použít přímo aplikací:

- Služba WCF musí být spuštěna dříve, než se spustí nástroj svcutil.
- Služba WCF musí vystavit svá metadata pomocí portu HTTP navíc k odkazům na koncový bod vlastního kanálu produktu IBM MQ pro generování klienta přímo ze spuštěné služby.
- Vlastní kanál musí být registrován v konfiguračních datech pro svcutil.

Informace o této úloze

Následující kroky vysvětlují, jak generovat klienta pro službu, která je nakonfigurována pro použití vlastního kanálu produktu IBM MQ , ale také odkrývá svá metadata za běhu prostřednictvím samostatného portu HTTP:

1. Spusťte službu WCF (služba musí být spuštěna dříve, než se spustí nástroj svcutil).
2. Přidejte podrobnosti z konfiguračního souboru `svcutil.exe` z kořenového adresáře instalace do aktivního konfiguračního souboru svcutil, zpravidla `C:\Program Files\Microsoft SDKs\Windows\v6.0A\bin\svcutil.exe.config` , takže svcutil rozpozná vlastní kanál IBM MQ .
3. Spusťte svcutil z příkazového řádku, například:

```
svcutil /language:C# /r: installlocation\bin\IBM.XMS.WCF.dll
/config:app.config http://localhost:8000/IBM.XMS.WCF/samples
```

4. Zkopírujte vygenerované soubory `app.config` a `YourService.cs` do projektu klienta produktu Microsoft Visual studio.

Jak pokračovat dále

Pokud nelze metadata služeb přímo načíst, lze místo toho použít svcutil ke generování souborů klienta z wsdl. Další informace viz: “Generování serveru proxy klienta WCF a konfiguračních souborů aplikace pomocí nástroje svcutil s kódem WSDL” na stránce 1240

Generování serveru proxy klienta WCF a konfiguračních souborů aplikace pomocí nástroje svcutil s kódem WSDL

Pokyny pro generování klientů WCF z WSDL, pokud metadata služby nejsou k dispozici.

1. Přidejte následující definice oboru názvů a informace o zásadě:

```
<wsdl:definitions
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
utility-1.0.xsd">

        <wsp:Policy wsu:Id="CustomBinding_IWMQSampleContract_policy">
            <wsp:ExtactlyOne>
                <wsp:All>
                    <xms:xms xmlns:xms="http://sample.schemas.ibm.com/policy/xms"/>
                </wsp:All>
            </wsp:ExactlyOne>
        </wsp:Policy>

...

</wsdl:definitions>
```

2. Upravte sekci vazeb tak, aby odkazovala na novou sekci zásady, a odeberte definici produktu `transport` ze základního prvku vazby:

```
<wsdl:definitions ...>

        <wsdl:binding ...>
                <wsp:PolicyReference URI="#CustomerBinding_IWMQSampleContract_policy"/>
                <[soap]:binding ... transport=""/>
                ...
        </wsdl:binding>
</wsdl:definitions>
```

3. Spusťte svcutil z příkazového řádku, například:

```
svcutil /language:C# /r:  MQ_INSTALLATION_PATH\bin\IBM.XMS.WCF.dll
/config:app.config  MQ_INSTALLATION_PATH\src\samples\WMQAxis\default\service
\soap.server.stockQuoteAxis_Wmq.wsdl
```

Sestavování aplikací klienta WCF pomocí serveru proxy klienta s konfiguračním souborem aplikace

Než začnete

Vytvořte nebo upravte konfigurační soubor aplikace pro klienta, jak je popsáno v tématu: “Vytvoření vlastního uživatelského kanálu WCF administrativně dodáním informací o vazbách a koncových bodech v konfiguračním souboru aplikace” na stránce 1230

Informace o této úloze

Vytvořit instanci a otevřít instanci serveru proxy klienta. Parametr předaný generovanému serveru proxy musí být stejný jako název koncového bodu určený v konfiguračním souboru klienta, například `Endpoint_WMQ`:

```
MyClientProxy myClient = new MyClientProxy("Endpoint_WMQ");
            try {
                   myClient.myMethod("HelloWorld!");
                   myClient.Close();
            }
            catch (TimeoutException e) {
                Console.Out.WriteLine(e);
                myClient.Abort();
            }
            catch (CommunicationException e) {
                Console.Out.WriteLine(e);
                myClient.Abort();
            }
            catch (Exception e) {
                Console.Out.WriteLine(e);
                myClient.Abort();
            }
```

Sestavování aplikací klienta WCF s použitím serveru proxy klienta s programovou konfigurací

Než začnete

1. Přidejte odkaz na soubor vlastního kanálu `IBM.XMS.WCF.dll` do projektu. `IBM.XMS.WCF.dll` je v adresáři *`WMQInstallDir`*`\bin` , kde *WMQInstallDir* je adresář, ve kterém je nainstalován produkt IBM MQ .
2. Přidejte příkaz *using* do oboru názvů `IBM.XMS.WCF` , například: `using IBM.XMS.WCF`
3. Vytvořte instanci prvku vazby a koncový bod kanálu, jak je popsáno v tématu: “Vytvoření vlastního kanálu WCF nahrazením informací o vazbách a koncovém bodu programově” na stránce 1232

Informace o této úloze

Pokud jsou vyžadovány změny vlastností vazby kanálu, proveďte následující kroky.

1. Přidejte vlastnosti vazby do produktu `transportBindingElement` , jak ukazuje následující obrázek:

```
SoapJmsIbmTransportBindingElement transportBindingElement = new
SoapJmsIbmTransportBindingElement();
Binding binding = new CustomBinding(new TextMessageEncodingBindingElement(),
transportBindingElement);
EndpointAddress address =
  new EndpointAddress("jms:/queue?destination=SampleQ@QM1&connectionFactory=
     connectQueueManager(QM1)&initialContextFactory=com.ibm.mq.jms.Nojndi");
```

2. Vytvořte proxy klienta, jak ukazuje následující obrázek, kde *binding* a *endpoint address* jsou vazba a adresa koncového bodu konfigurované v kroku 1 a předaná v:

```
 MyClientProxy myClient = new MyClientProxy(binding, endpoint address);
            try {
```

```
                myClient.myMethod("HelloWorld!");
                myClient.Close();
            }
            catch (TimeoutException e) {
                Console.Out.WriteLine(e);
                myClient.Abort();
            }
            catch (CommunicationException e) {
                Console.Out.WriteLine(e);
                                myClient.Abort();
            }
            catch (Exception e) {
                Console.Out.WriteLine(e);
                myClient.Abort();
            }
```

Použití ukázek WCF

Ukázky produktu Windows Communication Foundation (WCF) poskytují některé jednoduché příklady použití vlastního kanálu produktu IBM MQ .

Chcete-li sestavit ukázkové projekty, je třeba použít buď Microsoft.NET 3.5 SDK, nebo Microsoft Visual Studio 2008.

Ukázka klienta WCF jednosměrného klienta a serveru

Tato ukázka demonstruje použití vlastního kanálu produktu IBM MQ používaného ke spuštění služby WCF (Windows Communication Foundation) z klienta WCF pomocí tvaru jednosměrného kanálu.

Informace o této úloze

Služba implementuje jedinou metodu, jejíž výstupem je řetězec na konzolu. Klient byl generován pomocí nástroje `svcutil` k načtení metadat služby ze samostatně vystaveného koncového bodu HTTP, jak je popsáno v tématu "Generování serveru proxy klienta WCF a konfiguračních souborů aplikace pomocí nástroje svcutil s metadaty ze spuštěné služby" na stránce 1239 .

Ukázka byla nakonfigurována se specifickými názvy prostředků, jak je popsáno v následujícím postupu. Musíte-li změnit názvy prostředků, musíte také změnit odpovídající hodnotu v klientské aplikaci v souboru *MQ_INSTALLATION_PATH* `\tools\dotnet\samples\cs\wcf\samples\WCF\oneway\client\app.config` a v aplikaci služeb v souboru *MQ_INSTALLATION_PATH* `\tools\dotnet\samples\cs\wcf\samples\WCF\oneway\service\TestServices.cs` , kde *MQ_INSTALLATION_PATH* je instalační adresář pro IBM MQ. Další informace o formátování identifikátoru URI koncového bodu produktu JMS naleznete v tématu *Přenos protokolu SOAP produktu IBM MQ* v dokumentaci produktu IBM MQ . Potřebujete-li upravit ukázkové řešení a zdroj, pak potřebujete prostředí IDE, například Microsoft Visual Studio 8 nebo vyšší.

Postup

1. Vytvořte správce front s názvem *QM1* .
2. Vytvořte místo určení fronty s názvem *SampleQ* .
3. Spusťte službu tak, aby modul listener čekal na zprávy: Spusťte soubor *MQ_INSTALLATION_PATH* `\tools\dotnet\samples\cs\wcf\samples\WCF\oneway\service\bin\Release\TestService.exe` , kde *MQ_INSTALLATION_PATH* je instalační adresář pro IBM MQ.
4. Spusťte klienta jednou: Spusťte soubor *MQ_INSTALLATION_PATH* `\tools\dotnet\samples\cs\wcf\samples\WCF\oneway\client\bin\Release\TestClient.exe` , kde *MQ_INSTALLATION_PATH* je instalační adresář pro IBM MQ.

   Klientská aplikační smyčka pětkrát odesílá pět zpráv do *SampleQ* .

Výsledky

Aplikace služeb získává zprávy z *SampleQ* a na obrazovce se zobrazí `Hello World` pětkrát.

Jak pokračovat dále

Jednoduchý požadavek-klient odpovědí a server WCF serveru

Tato ukázka demonstruje vlastní kanál produktu IBM MQ používaný ke spuštění služby WCF (Windows Communication Foundation) z klienta WCF s použitím tvaru kanálu s odpovědí typu požadavek-odezva.

Informace o této úloze

Tato služba poskytuje některé jednoduché metody kalkulačky k přidání a odečtení dvou čísel a následné vrácení výsledku. Klient byl generován pomocí nástroje `svcutil` k načtení metadat služby ze samostatně vystaveného koncového bodu HTTP, jak je popsáno v tématu “Generování serveru proxy klienta WCF a konfiguračních souborů aplikace pomocí nástroje svcutil s metadaty ze spuštěné služby” na stránce 1239 .

Ukázka byla nakonfigurována se specifickými názvy prostředků, jak je popsáno v následujícím postupu. Potřebujete-li změnit názvy prostředků, musíte také změnit odpovídající hodnotu v klientské aplikaci v souboru *MQ_INSTALLATION_PATH* `\Tools\wcf\samples\WCF\requestreply\client\app.config` a v aplikaci služeb v souboru *MQ_INSTALLATION_PATH* `\Tools\wcf\samples\WCF\requestreply\service\RequestReplyService.cs` , kde *MQ_INSTALLATION_PATH* je instalační adresář pro IBM MQ. Další informace o formátování identifikátoru URI koncového bodu produktu JMS naleznete v tématu *Přenos protokolu SOAP produktu IBM MQ* v dokumentaci produktu IBM MQ . Potřebujete-li upravit ukázkové řešení a zdroj, pak potřebujete prostředí IDE, například Microsoft Visual Studio 8 nebo vyšší.

Postup

1. Vytvořte správce front s názvem *QM1* .
2. Vytvořte místo určení fronty s názvem *SampleQ* .
3. Vytvořte cíl fronty s názvem *SampleReplyQ* .
4. Spusťte službu tak, aby modul listener čekal na zprávy: Spusťte soubor *MQ_INSTALLATION_PATH* `\Tools\wcf\samples\WCF\requestreply\service\bin\Release\SimpleRequestReply_Service.exe` , kde *MQ_INSTALLATION_PATH* je instalační adresář pro IBM MQ.
5. Spusťte klienta jednou: Spusťte soubor *MQ_INSTALLATION_PATH* `\Tools\wcf\samples\WCF\requestreply\client\bin\Release\SimpleRequestReply_Client.exe` , kde *MQ_INSTALLATION_PATH* je instalační adresář pro IBM MQ.

Výsledky

Když je klient spuštěn, spustí se následující proces a opakuje se čtyřikrát, takže se každý z pěti zpráv posílá každý:

1. Klient vloží zprávu s požadavkem na adresu *SampleQ* a čeká na odpověď.
2. Služba získá zprávu požadavku z adresáře *SampleQ*.
3. Služba přidá a odečítá některé hodnoty pomocí obsahu zprávy.
4. Služba pak vloží výsledky do zprávy v *SampleReplyQ*a čeká, až klient vloží novou zprávu.
5. Klient získá zprávu z fronty *SampleReplyQ* a zobrazí výsledky na obrazovce.

Jak pokračovat dále

Klient WCF do služby .NET hostované ukázkou produktu IBM MQ

Ukázkové klientské aplikace a ukázkové aplikace serveru proxy služeb jsou dodávány pro produkt .NET i pro produkt Java. Ukázky jsou založeny na službě Stock Quote, která přijímá požadavek na akcie akcií a poté poskytuje kótování akcií.

Než začnete

Ukázka vyžaduje, aby bylo prostředí produktu .NET SOAP over JMS hostujícího prostředí správně nainstalováno a nakonfigurováno v produktu IBM MQ a je přístupné z lokálního správce front. Chcete-li získat informace o instalaci a konfiguraci prostředí, prohlédněte si: "Instalace produktu IBM MQ Web Transport pro SOAP" na stránce 1265

When the .NET SOAP over JMS service hosting environment is correctly installed and configured in IBM MQ and is accessible from a local queue manager, additional configuration steps must be completed.

1. Nastavte proměnnou prostředí WMQSOAP_HOME na instalační adresář produktu IBM MQ , například: `C:\Program Files\IBM\MQ`
2. Ujistěte se, že je kompilátor Java `javac` dostupný a v proměnné PATH.
3. Zkopírujte soubor `axis.jar` z adresáře `prereqs/axis` instalačního disku CD produktu WebSphere do produkčního adresáře produktu IBM MQ , například: `C:\Program Files\IBM\MQ\java\lib\soap`
4. Přidejte do proměnné PATH: *MQ_INSTALLATION_PATH*`\Java\lib` , kde *MQ_INSTALLATION_PATH* představuje adresář, kde je nainstalován produkt IBM MQ , například: `C:\Program Files\IBM\MQ`
5. Ujistěte se, že umístění .NET je zadáno správně v *MQ_INSTALLATION_PATH*`\bin\amqwcallWSDL.cmd` , kde *MQ_INSTALLATION_PATH* představuje adresář, kde je nainstalován produkt IBM MQ , například: `C:\Program Files\IBM\MQ`. Umístění .NET může být zadáno například: `set msfwdir=%ProgramFiles%\Microsoft Visual Studio .NET 2003\SDK\v1.1\Bin`

Když jsou předchozí kroky dokončeny, otestujte a spusťte službu:

1. Přejděte do pracovního adresáře protokolu SOAP prostřednictvím produktu JMS .
2. Zadejte jeden z následujících příkazů pro spuštění testu ověření a nechte spuštěný modul listener služby:
   - Pro .NET: *MQ_INSTALLATION_PATH*`\Tools\soap\samples\runivt dotnet hold` , kde *MQ_INSTALLATION_PATH* představuje adresář, kde je nainstalován produkt IBM MQ .
   - Pro AXIS: *MQ_INSTALLATION_PATH*`\Tools\soap\samples\runivt Dotnet2AxisClient hold` kde *MQ_INSTALLATION_PATH* představuje adresář, kde je nainstalován produkt IBM MQ .

Argument `hold` uchovává moduly listener spuštěné po dokončení testu.

Pokud jsou během této konfigurace ohlášeny chyby, můžete odebrat všechny změny, aby bylo možné proceduru restartovat následujícím způsobem:

1. Odstraňte vygenerovaný adresář SOAP over JMS .
2. Odstraňte správce front.

Informace o této úloze

Tato ukázka předvádí připojení klienta WCF k ukázkové službě produktu .NET SOAP over JMS , která je poskytována v produktu IBM MQ , a to pomocí tvaru jednosměrného kanálu. Služba implementuje jednoduchý příklad StockQuote , který výstupem je textový řetězec na konzolu.

Klient byl vygenerován pomocí jazyka WSDL ke generování souborů klienta, jak je popsáno v tématu "Generování serveru proxy klienta WCF a konfiguračních souborů aplikace pomocí nástroje svcutil s kódem WSDL" na stránce 1240 .

Ukázka byla nakonfigurována se specifickými názvy prostředků, jak je popsáno v následujícím postupu. Potřebujete-li změnit názvy prostředků, musíte také změnit odpovídající hodnotu v klientské aplikaci v souboru *MQ_INSTALLATION_PATH* `\tools\wcf\samples\WMQNET\default\client\app.config` a na aplikaci služby v souboru *MQ_INSTALLATION_PATH* `\tools\wcf\samples\WMQNET\default\service\WmqDefaultSample_StockQuoteDotNet.wsdl` , kde *MQ_INSTALLATION_PATH* představuje instalační adresář pro IBM MQ. Další informace o formátování identifikátoru URI koncového bodu produktu JMS naleznete v tématu *Přenos protokolu SOAP produktu IBM MQ* v dokumentaci produktu IBM MQ .

Postup

Spustit klienta jednou: Spusťte soubor *MQ_INSTALLATION_PATH* `\tools\wcf\samples\WMQNET\default\client\bin\Release\TestClient.exe` , kde *MQ_INSTALLATION_PATH* představuje instalační adresář pro IBM MQ.

Klientská aplikační smyčka pětkrát odesílá pět zpráv do ukázkové fronty.

Výsledky

Aplikace služby získá zprávy z ukázkové fronty a zobrazí `Hello World` pětkrát na obrazovce.

Klient WCF do služby Axis Java hostované ukázkou produktu IBM MQ

Ukázkové klientské aplikace a ukázkové aplikace serveru proxy služeb jsou dodávány pro produkt Java i pro produkt .NET. Ukázky jsou založeny na službě Stock Quote, která přijímá požadavek na akcie akcií a poté poskytuje kótování akcií.

Než začnete

Tento vzorek vyžaduje, aby bylo prostředí produktu .NET SOAP over JMS hosting Environment správně nainstalováno a nakonfigurováno v produktu IBM MQ a je přístupné z lokálního správce front. Chcete-li získat informace o instalaci a konfiguraci prostředí, prohlédněte si: “Instalace produktu IBM MQ Web Transport pro SOAP” na stránce 1265

When the .NET SOAP over JMS service hosting environment is correctly installed and configured in IBM MQ and is accessible from a local queue manager, additional configuration steps must be completed.

1. Nastavte proměnnou prostředí WMQSOAP_HOME na instalační adresář produktu IBM MQ , například: `C:\Program Files\IBM\MQ`
2. Ujistěte se, že je kompilátor Java `javac` dostupný a v proměnné PATH.
3. Zkopírujte soubor `axis.jar` z adresáře `prereqs/axis` instalačního disku CD produktu WebSphere do instalačního adresáře produktu IBM MQ .
4. Přidejte do proměnné PATH: *MQ_INSTALLATION_PATH*`\Java\lib` , kde *MQ_INSTALLATION_PATH* představuje adresář, kde je nainstalován produkt IBM MQ , například: `C:\Program Files\IBM\MQ`
5. Ujistěte se, že umístění .NET je zadáno správně v *MQ_INSTALLATION_PATH*`\bin\amqwcallWSDL.cmd` , kde *MQ_INSTALLATION_PATH* představuje adresář, kde je nainstalován produkt IBM MQ , například: `C:\Program Files\IBM\MQ`. Umístění .NET může být zadáno například: `set msfwdir=%ProgramFiles%\Microsoft Visual Studio .NET 2003\SDK\v1.1\Bin`

Když jsou předchozí kroky dokončeny, otestujte a spusťte službu:

1. Přejděte do pracovního adresáře protokolu SOAP prostřednictvím produktu JMS .
2. Zadejte jeden z následujících příkazů pro spuštění testu ověření a nechte spuštěný modul listener služby:
   - Pro .NET: *MQ_INSTALLATION_PATH*`\Tools\soap\samples\runivt dotnet hold` , kde *MQ_INSTALLATION_PATH* představuje adresář, kde je nainstalován produkt IBM MQ .
   - Pro AXIS: *MQ_INSTALLATION_PATH*`\Tools\soap\samples\runivt Dotnet2AxisClient hold` kde *MQ_INSTALLATION_PATH* představuje adresář, kde je nainstalován produkt IBM MQ .

Argument `hold` uchovává moduly listener spuštěné po dokončení testu.

Pokud jsou během této konfigurace ohlášeny chyby, můžete odebrat všechny změny tak, aby se procedura restartovala následujícím způsobem:

1. Odstraňte vygenerovaný adresář SOAP over JMS .
2. Odstraňte správce front.

Informace o této úloze

Ukázka demonstruje připojení z klienta WCF k ukázkové službě SOAP Java Axis over JMS , která je poskytována v produktu IBM MQ pomocí jednosměrného tvaru kanálu. Služba implementuje jednoduchý příklad StockQuote , který posílá textový řetězec do souboru, který je uložen v aktuálním adresáři.

Klient byl vygenerován pomocí jazyka WSDL ke generování souborů klienta, jak je popsáno v tématu “Generování serveru proxy klienta WCF a konfiguračních souborů aplikace pomocí nástroje svcutil s kódem WSDL” na stránce 1240 .

Ukázka byla nakonfigurována se specifickými názvy prostředků, jak je popsáno v tomto odstavci. Potřebujete-li změnit názvy prostředků, musíte také změnit odpovídající hodnotu v klientské aplikaci v souboru *MQ_INSTALLATION_PATH* `\tools\wcf\samples\WMQAxis\default\client\app.config` a na aplikaci služby v souboru *MQ_INSTALLATION_PATH* `\tools\wcf\samples\WMQAxis\default\service\WmqDefaultSample_StockQuoteDotNet.wsdl` , kde *MQ_INSTALLATION_PATH* představuje instalační adresář pro IBM MQ.

Postup

Spustit klienta jednou: Spusťte soubor *MQ_INSTALLATION_PATH* `\tools\wcf\samples\WMQAxis\default\client\bin\Release\TestClient.exe` , kde *MQ_INSTALLATION_PATH* představuje instalační adresář pro IBM MQ.

Klientská aplikační smyčka pětkrát odesílá pět zpráv do ukázkové fronty.

Výsledky

Aplikace služby získá zprávy z ukázkové fronty a přidá `Hello World` pětkrát do souboru v aktuálním adresáři.

Související odkazy

“Ošetřování různých názvů prvků odezvy SOAP” na stránce 1255

WCF očekává, že název navrácené hodnoty bude standardně ve specifickém formátu, ale služba nemusí vrátit prvek s jeho jménem v očekávaném formátu.

Klient WCF do služby Java , jehož hostitelem je ukázka WebSphere Application Server

Ukázkové klientské aplikace a ukázkové aplikace proxy služeb jsou dodávány pro produkt WebSphere Application Server 6. Poskytne se také služba požadavek-odezva.

Než začnete

Tato ukázka vyžaduje použití následující konfigurace produktu IBM MQ :

Tabulka 183. IBM MQ povinná konfigurace

Objekt	**Povinné jméno**
Správce front	QM1
Lokální fronta	HelloWorld
Lokální fronta	Odpověď HelloWorld

Tato ukázka také vyžaduje, aby bylo prostředí hostitele produktu WebSphere Application Server 6 správně nainstalováno a nakonfigurováno. Produkt WebSphere Application Server 6 používá při výchozím nastavení připojení režimu vazeb k produktu IBM MQ . Produkt WebSphere Application Server 6 proto musí být nainstalován na stejném počítači jako správce front.

Po konfiguraci prostředí WAS musí být dokončeny následující další kroky konfigurace:

1. Vytvořte následující objekty JNDI v úložišti JNDI produktu WebSphere Application Server :

a. Místo určení fronty JMS s názvem `HelloWorld`
   - Nastavte název rozhraní JNDI na hodnotu `jms/HelloWorld` .
   - Nastavit název fronty na `HelloWorld`

b. Továrna připojení fronty produktu JMS s názvem `HelloWorldQCF`
   - Nastavte název rozhraní JNDI na hodnotu `jms/HelloWorldQCF` .
   - Nastavit název správce front na `QM1`

c. Továrna připojení fronty produktu JMS s názvem `WebServicesReplyQCF`
   - Nastavte název rozhraní JNDI na hodnotu `jms/WebServicesReplyQCF` .
   - Nastavit název správce front na `QM1`

2. Vytvořte port modulu listener pro zprávy s názvem `HelloWorldPort` v produktu WebSphere Application Server s následující konfigurací:
   - Nastavte název rozhraní JNDI továrny připojení na `jms/HelloWorldQCF`
   - Nastavte název rozhraní JNDI místa určení na hodnotu `jms/HelloWorld` .
3. Nainstalujte aplikaci webové služby `HelloWorldEJBEAR.ear` na aplikační server WebSphere následujícím způsobem:
   a. Klepněte na volbu **Aplikace** > **Nová aplikace** > **Nová podniková aplikace**.
   b. Přejděte do adresáře *MQ_INSTALLATION_PATH*`\tools\wcf\samples\WAS\HelloWorldsEJBEAR.ear` , kde *MQ_INSTALLATION_PATH* je instalační adresář produktu IBM MQ.
   c. Neměňte žádnou výchozí volbu v průvodci a restartujte aplikační server po instalaci aplikace.

Po dokončení konfigurace serveru WAS ji otestujte spuštěním této služby:

1. Přejděte do pracovního adresáře produktu Soap over JMS .
2. Zadejte tento příkaz ke spuštění ukázky: *MQ_INSTALLATION_PATH* `\tools\wcf\samples\WAS\TestClient.exe` , kde *MQ_INSTALLATION_PATH* je instalační adresář produktu IBM MQ.

Informace o této úloze

Ukázka demonstruje připojení z klienta WCF k ukázkové službě produktu WebSphere Application Server SOAP over JMS , která je součástí ukázek WCF zahrnutých v produktu IBM MQ, pomocí tvaru kanálu požadavek-odezva. Tok zpráv mezi WCF a serverem WebSphere Application Server pomocí front produktu IBM MQ . Služba implementuje metodu `HelloWorld(...)` , která přijímá řetězec a vrací pozdrav klientovi.

Klient byl generován pomocí nástroje svcutil k načtení metadat služby ze samostatně vystaveného koncového bodu HTTP, jak je popsáno v tématu “Generování serveru proxy klienta WCF a konfiguračních souborů aplikace pomocí nástroje svcutil s metadaty ze spuštěné služby” na stránce 1239

Ukázka byla nakonfigurována se specifickými názvy prostředků, jak je popsáno v následujícím postupu. Potřebujete-li změnit názvy prostředků, musíte také změnit odpovídající hodnotu v klientské aplikaci v souboru *MQ_INSTALLATION_PATH* `\tools\wcf\samples\WAS\default\client\app.config` a v aplikaci služby v produktu *MQ_INSTALLATION_PATH* `\tools\wcf\samples\WAS\HelloWorldsEJBEAR.ear` , kde *MQ_INSTALLATION_PATH* je instalační adresář produktu IBM MQ. Další informace o formátování identifikátoru URI koncového bodu produktu JMS naleznete v tématu Syntaxe identifikátoru URI a parametry pro implementaci webové služby.

Služba a klient jsou založeny na službě a klientovi nastíněné v článku IBM Developer *Sestavení webové služby JMS pomocí protokolu SOAP prostřednictvím produktů JMS a WebSphere Studio*. Další informace o vývoji webových služeb SOAP prostřednictvím produktu JMS , které jsou kompatibilní s vlastním kanálem WCF produktu IBM MQ , naleznete na adrese https://www.ibm.com/developerworks/websphere/library/techarticles/0402_du/0402_du.html.

Postup

Spusťte klienta jednou: Spusťte soubor *MQ_INSTALLATION_PATH* `\tools\wcf\samples\WAS\default\client\bin\Release\TestClient.exe`, kde *MQ_INSTALLATION_PATH* je instalační adresář pro IBM MQ.

Klientská aplikace spouští současně obě metody služby a odesílá dvě zprávy do ukázkové fronty.

Výsledky

Aplikace služeb získává zprávy z ukázkové fronty a poskytuje odezvu na metodu `HelloWorld(...)`, kterou jsou výstupem aplikace klienta na konzolu.

Určování problémů s vlastním kanálem WCF pro produkt IBM MQ

Pomocí trasování produktu IBM MQ můžete shromažďovat podrobné informace o tom, které různé části kódu IBM MQ dělají. Při použití produktu Windows Communication Foundation (WCF) je generován samostatný výstup trasování pro trasování vlastního kanálu WCF, který je integrován s trasováním infrastruktury WCF produktu Microsoft .

Úplné zpřístupňování trasování pro vlastní kanál WCF produkuje dva výstupní soubory:

1. Vlastní trasování kanálu WCF bylo integrováno s trasováním infrastruktury WCF produktu Microsoft .
2. Vlastní trasování kanálu WCF bylo integrováno s XMS .NET.

Pokud máte dva výstupy trasování, problémy lze sledovat na každém rozhraní pomocí příslušných nástrojů, například:

- Určování problémů se WCF pomocí vhodných nástrojů produktu Microsoft .
- Problémy produktu IBM MQ MQI client s použitím trasovacího formátu XMS .

Chcete-li zjednodušit povolení trasování, trasovací zásobník .NET 3 TraceSource a XMS .NET je řízen pomocí jednoho rozhraní, jak je popsáno v: “Konfigurace trasování a trasování souborů trasování WCF: Rozhraní SOAP/JMS” na stránce 1249.

Hierarchie výjimek vlastního kanálu WCF

Typy výjimek vyvolané vlastním kanálem jsou konzistentní s WCF a obvykle se jedná o výjimku TimeoutException nebo CommunicationException (nebo podtřídy CommunicationException). Další podrobnosti o chybovém stavu, jsou-li k dispozici, jsou poskytovány pomocí propojených nebo vnitřních výjimek.

Rozhraní SOAP/JMS

Následující výjimky jsou typickými příklady a každá vrstva v architektuře kanálu přispívá k další propojené výjimce, například CommunicationsException má propojenou výjimku XMSException, která má propojenou výjimku MQException:

1. System.serviceModel.CommunicationsExceptions
2. IBM.XMS.XMSException
3. IBM.WMQ.MQException

Informace o klíči jsou zachyceny a poskytnuty v kolekci dat nejvyšší CommunicationException v hierarchii. Zachycení a zajištění dat zabraňuje v tom, aby aplikace propojují s každou vrstvou v architektuře kanálu, aby bylo možné vyslýchat propojené výjimky a všechny další informace, které mohou obsahovat. Jsou definovány následující názvy kláves:

- IBM.XMS.WCF.ErrorCode: Kód chybové zprávy o aktuální výjimce vlastního kanálu.
- IBM.XMS.ErrorCode: Chybová zpráva první výjimky XMS v zásobníku.
- IBM.WMQ.ReasonCode: Kód příčiny IBM MQ .
- IBM.WMQ.CompletionCode: Základní kód dokončení IBM MQ .

Rozhraní Non-SOAP/Non-JMS

Následující výjimky jsou typickými příklady a každá vrstva v architektuře kanálu přispívá k další propojené výjimce, například výjimka CommunicationsException má propojenou výjimku MQException:

1. System.ServiceModel.CommunicationsExceptions
2. IBM.WMQ.MQException

Informace o klíči jsou zachyceny a poskytnuty v kolekci dat nejvyšší CommunicationException v hierarchii. Zachycení a zajištění dat zabraňuje v tom, aby aplikace propojují s každou vrstvou v architektuře kanálu, aby bylo možné vyslýchat propojené výjimky a všechny další informace, které mohou obsahovat. Jsou definovány následující názvy kláves:

- IBM.WMQ.WCF.ErrorCode: Kód chybové zprávy o aktuální výjimce vlastního kanálu.
- IBM.WMQ.ReasonCode: Kód příčiny IBM MQ .
- IBM.WMQ.CompletionCode: Základní kód dokončení IBM MQ .

Konfigurace trasování WCF

Pro konfiguraci trasování WCF jsou k dispozici dvě volby. Trasování můžete buď konfigurovat programově, nebo prostřednictvím proměnné prostředí.

Konfigurace trasování a trasování souborů trasování WCF: Rozhraní SOAP/JMS

Je-li trasování plně povoleno, vytvoří dva výstupní soubory, jeden pro diagnostiku problémů WCF a jeden podrobný soubor pro vnitřní diagnostický materiál pro trasování. Chcete-li zjednodušit povolení trasování, trasovací sady produktu .NET 3 TraceSource a XMS .NET používají jedno rozhraní.

Pro vlastní kanál WCF jsou k dispozici dvě různé metody trasování. Tyto dvě metody trasování jsou aktivovány nezávisle nebo společně. Každá metoda vytvoří svůj vlastní trasovací soubor, takže když jsou obě metody trasování aktivovány, vygenerují se dva výstupní soubory trasování.

Chcete-li udržet konfiguraci a povolení co nejjednodušší, použije se stejné rozhraní k řízení obou metod trasování. Soubor `app.config` musí být upraven tak, aby zahrnoval příslušnou konfiguraci trasování, jak je popsáno v následující sekci. Uživatelé pak mohou přidávat své vlastní ekvivalentní sekce ke sloučení výstupu s trasováním ze své vlastní aplikace.

Vlastní trasování kanálu WCF není ve výchozím nastavení povoleno. Nejprve je třeba vytvořit modul listener pro trasování a poté nastavit požadovanou úroveň trasování pro vybraný zdroj trasování v souboru `app.config` .

Konfigurace vlastního kanálu WCF s trasováním infrastruktury WCF

Přidejte následující sekci kódu do sekce `<system.diagnostics><sources>` v souboru `app.config` :

```
<source name="IBM.XMS.WCF" switchValue="Verbose,ActivityTracing">
    <listeners>
        <remove name="Default"/>
        <add name="NewListener"/>
    </listeners>
</source>
```

Předchozí část kódu vytváří trasování kanálu pomocí .NET 3 TraceSource. Všechna vyvolání konfiguračních souborů přidružených ke spustitelným souborům jsou řízena touto částí kódu.

Konfigurace vlastního kanálu WCF s trasováním XMS .NET

Konfigurace trasování produktu XMS .NET vyžaduje, abyste přidali sekci kódu do sekce `<system.diagnostics><sources>` v souboru `app.config` . Část kódu se však přidá do rozšiřitelného prvku `<source>` zobrazeného v sekci Konfigurace vlastního kanálu WCF s trasováním infrastruktury WCF . Takže ačkoli se musí nacházet trasovací kód infrastruktury WCF pro trasování XMS .NET , trasování

infrastruktury WCF může být vypnuto, pokud není vyžadováno, jak je popsáno v části Povolení trasování WCF .

```
<source name="IBM.XMS.WCF" switchValue="Verbose, ActivityTracing"
 xmsTraceSpecification="*=all=enabled" xmsTraceFilePath="path"
 xmsTraceFileSize="2000000" xmsTraceFileNumber="4" xmsTraceFormat="advanced">
    <listeners>
        <remove name="Default"/>
        <add name="NewListener"/>
    </listeners>
</source>
```

Konfigurační proměnné trasování WCF

Tabulka 184. Konfigurační proměnné trasování WCF

Proměnná	**Popis**
název	Zadejte název jako: `IBM.XMS.WCF`
switchValue	Úroveň trasování řídí switchValue . Je-li parametr switchValue nastaven na hodnotu `Off`, infrastruktura WCF TraceSource se nevygeneruje. Jakákoli jiná hodnota, jako např. `Verbose`, vygeneruje TraceSource. Podrobné informace o úrovni trasování z produktu Microsoftnaleznete v dokumentaci k produktu WCF nebo na webové stránce Microsoft WCF Tracing na adrese: https://msdn.microsoft.com/en-us/library/ms733025(vs.85).aspx

xmsTraceSpecifikace = *ComponentName* = *type* = *state*	*ComponentName* je název třídy, kterou chcete trasovat. V tomto názvu můžete použít zástupný znak *. Příklad:
	`*=all=enabled`
	určuje, že chcete trasovat všechny třídy, a
	`IBM.XMS.impl.*=all=enabled`
	uvádí, že požadujete pouze trasování rozhraní API.
	Typ *typ* může být libovolný z následujících typů trasování:
	• vše
	• ladění
	• událost
	• EntryExit
	Stav *stav* může být povolen nebo zakázán.

Tabulka 184. Konfigurační proměnné trasování WCF (pokračování)	
Proměnná	**Popis**
xmsTraceFilePath= "*název_souboru*"	Pokud neuvedete xmsTraceFilePath, nebo pokud je xmsTraceFilePath přítomen, ale obsahuje prázdný řetězec, pak trasovací soubor se umístí do aktuálního adresáře. Chcete-li uložit trasovací soubor do uvedeného adresáře, zadejte v souboru xmsTraceFilePathnázev adresáře, například: `xmsTraceFilePath="c:\somepath"`
xmsTraceFileSize= "*velikost*"	Maximální povolená velikost trasovacího souboru. Když soubor dosáhne této velikosti, je archivován a přejmenován. Výchozí maximum je 20 KB, což je uvedeno jako: `xmsTraceFileSize="20000000".`
xmsTraceFileNumber= "*číslo*"	Počet trasovacích souborů, které mají být uchovány. Předvolba je 4 (jeden aktivní soubor a tři archivní soubory). Minimální povolený počet je dva.

xmsTraceFormat="*formát*"	Existují dvě úrovně formátu xmsTrace: `basic` a `advanced`. Výchozí formát trasování je základní, pokud nezadáte formát xmsTrace, nebo pokud je přítomen formát xmsTrace, ale obsahuje prázdný řetězec. Trasovací soubory jsou vytvářeny v tomto formátu, pokud zadáte:
	`xmsTraceFormat="basic"`
	Pokud vyžadujete trasování, které je kompatibilní s nástroji pro analýzu trasování, musíte uvést:
	`traceFormat="advanced"`

Povolení trasování WCF

Pro povolení a zakázání dvou různých metod trasování jsou k dispozici čtyři kombinace. Tyto čtyři kombinace vyžadují úpravu hodnot oddílů kódu popsaných v předchozích sekcích.

Je zde také proměnná prostředí, kterou lze nastavit; další informace viz“Povolení trasování WCF pomocí proměnné prostředí WCF_TRACE_ON” na stránce 1252.

Tato tabulka a zobrazené hodnoty závisí na níže uvedených částech kódu, které již byly přidány do souboru `app.config` .

Tabulka 185. Kombinace povolení trasování WCF.

Typ trasování	Hodnota změněna	Příklad
Trasování XMS je povoleno. WCF TraceSource je povoleno	Hodnota `switchValue` není nastavena na hodnotu `Vypnuto` .	`<source name="IBM.XMS.WCF" switchValue=" Verbose, ActivityTracing" xmsTraceSpecification="*=all=enabled" xmsTraceFilePath="path" xmsTraceFileSize="2000000" xmsTraceFileNumber="4" xmsTraceFormat="advanced"> <listeners> <remove name="Default"/> <add name="NewListener"/> </listeners> </source>`
Trasování XMS je povoleno. WCF TraceSource je zakázaný	Parametr `switchValue` je nastaven na hodnotu `Vypnuto` a byl zadán `xmsTraceSpecification` .	`<source name="IBM.XMS.WCF" switchValue="Off, ActivityTracing" xmsTraceSpecification="*=all=enabled" xmsTraceFilePath="path" xmsTraceFileSize="2000000" xmsTraceFileNumber="4" xmsTraceFormat="advanced"> <listeners> <remove name="Default"/> <add name="NewListener"/> </listeners> </source>`
Trasování XMS je zakázáno. WCF TraceSource je povoleno	Tento výsledek lze dosáhnout dvěma způsoby: • Proměnná `switchValue` není nastavena na hodnotu `Off` a položka `xmsTraceSpecification` nebyla přidána. • Proměnná `switchValue` není nastavena na hodnotu `Off` a hodnota `xmsTraceSpecification` byla nastavena na hodnotu `disabled` .	`<source name="IBM.XMS.WCF" switchValue="Verbose, ActivityTracing" xmsTraceSpecification="*=all=disabled" xmsTraceFilePath="path" xmsTraceFileSize="2000000" xmsTraceFileNumber="4" xmsTraceFormat="advanced"> <listeners> <remove name="Default"/> <add name="NewListener"/> </listeners> </source>`
Trasování XMS je zakázáno. WCF TraceSource je zakázaný	Existují tři způsoby, jak dosáhnout tohoto výsledku: • V souboru `app.config` není žádný prvek `<source>` . • Proměnná `switchValue` je nastavena na hodnotu `Off` a položka `xmsTraceSpecification` nebyla přidána. • Proměnná `switchValue` je nastavena na hodnotu `Off` a parametr `xmsTraceSpecification` byl nastaven na hodnotu `disabled` .	`<source name="IBM.XMS.WCF" switchValue="Off, ActivityTracing" xmsTraceSpecification="*=all=disabled" xmsTraceFilePath="path" xmsTraceFileSize="2000000" xmsTraceFileNumber="4" xmsTraceFormat="advanced"> <listeners> <remove name="Default"/> <add name="NewListener"/> </listeners> </source>`

Povolení trasování WCF pomocí proměnné prostředí WCF_TRACE_ON

Stejně jako u předchozích metod popsaných pro povolení trasování WCF lze trasování XMS .NET povolit také pomocí proměnné prostředí WCF_TRACE_ON.

Nastavení proměnné prostředí WCF_TRACE_ON na jinou hodnotu než null je ekvivalentem nastavení hodnoty `xmstraceSpecification` na `*=all=enabled`, například: " `set WCF_TRACE_ON=true` "

Pokud je však parametr `xmstraceSpecification` explicitně nastaven v souboru `app.config` , bude přepsána proměnná prostředí WCF_TRACE_ON.

Výstupní soubory trasování WCF a názvy souborů

Trasovací soubory XMS jsou tradičně pojmenovány s použitím základního názvu a formátu ID procesu: `xms_trace_`*`pid`*`.log`, kde *pid* je ID procesu.

Protože mohou být trasovací soubory XMS stále produkovány paralelně s vlastními soubory trasování kanálu služby WCF, má vlastní trasování kanálu WCF integrované s trasovacími soubory trasování XMS .NET následující formát, aby nedošlo k záměně: `wcf xms_trace_`*`pid`*`.log`, kde *pid* je ID procesu.

Výstupní soubor trasování je při výchozím nastavení vytvořen v aktuálním pracovním adresáři, ale toto místo určení lze v případě potřeby předefinovat.

Konfigurace trasování WCF: Rozhraní Non_SOAP/Non-JMS

V případě rozhraní Non_SOAP/Non-JMS můžete buď konfigurovat trasování prostřednictvím proměnné prostředí, nebo programově.

Existují dva způsoby, jak povolit trasování pro rozhraní Non-SOAP/Non-JMS :

- Nastavením parametru WMQ_TRACE_ON jako proměnné prostředí.
- Přidáním následující části kódu do sekce `<system.diagnostics><sources>` v souboru app.config

```
<source name="IBM.WMQ.WCF" switchValue="Verbose, ActivityTracing"
xmsTraceSpecification="*=all=enabled"
xmsTraceFileSize="2000000" xmsTraceFileNumber="4"
xmsTraceFormat="advanced">
</source>
```

WCF XMS First Failure Support Technology (FFST)

Můžete shromažďovat podrobné informace o tom, které různé části kódu IBM MQ se provádí pomocí trasování produktu IBM MQ . XMS FFST má své vlastní konfigurační a výstupní soubory pro vlastní kanál WCF.

Trasovací soubory XMS FFST jsou tradičně pojmenovány pomocí základního názvu a formátu ID procesu: `xmsffdc` *`pid_date`*`.txt`, kde *pid* je ID procesu a *datum* je čas a datum.

Protože soubory trasování XMS FFST lze stále ještě vytvořit paralelně se soubory XMS FFST vlastního kanálu WCF, mají výstupní soubory vlastního kanálu WCF XMS FFST následující formát, aby nedošlo k záměně: `wcf ffdc` *`pid_date`*`.txt`, kde *pid* je ID procesu a *datum* je čas a datum.

Tento trasovací výstupní soubor je při výchozím nastavení vytvořen v aktuálním pracovním adresáři, ale toto místo určení lze v případě potřeby předefinovat.

Vlastní kanál WCF se záhlavím trasování XMS .NET je podobný jako v následujícím příkladu:

```
************* Start Display XMS WCF Environment *************
Product Name :- value
WCF Version  :- value
Level  :- value
************* End Display XMS WCF Environment *************
```

Trasovací soubory produktu FFST jsou formátovány standardním způsobem, bez formátování, které je specifické pro vlastní kanál.

Informace o verzi WCF

Informace o verzi WCF pomáhá při určování problémů a je zahrnuta v metadatech sestavení vlastního kanálu.

Vlastní kanál IBM MQ pro metadata verze WCF lze načíst jedním ze tří způsobů:

- Použití obslužného programu IBM MQ dspmqver. Informace o tom, jak používat příkaz dspmqver naleznete v následujícím tématu: dspmqver
- Pomocí dialogového okna vlastností Průzkumníka Windows : V Průzkumníku Windows klepněte pravým tlačítkem myši na **IBM.XMS.WCF.dll** > **Vlastnosti** > **Verze**.
- Ze záhlaví informace o všech kanálech FFST nebo trasovacích souborech. Další informace o informacích záhlaví produktu FFST naleznete v následujícím tématu: “WCF XMS First Failure Support Technology (FFST)” na stránce 1253

Rady a tipy WCF

Následující rady a tipy nejsou v žádném významném pořadí a mohou být přidány do té doby, kdy jsou uvolněny nové verze dokumentace. Jsou to témata, která vám mohou ušetřit čas, pokud jsou důležitá pro práci, kterou právě děláte.

Externalizace výjimek z hostitele služby WCF

Pro služby hostované pomocí hostitele služby WCF nejsou standardně externalizovány všechny neošetřené výjimky vyvolané službou, interními objekty WCF nebo zásobníkem kanálu. Chcete-li být informováni o těchto výjimkách, musí být registrována obslužná rutina chyb.

Následující kód poskytuje příklad definování chování služby obslužné rutiny chyb, které může být použito jako atribut služby:

```
using System.ServiceModel.Dispatcher;
using System.Collections.ObjectModel;
....
    public class ErrorHandlerBehaviorAttribute : Attribute, IServiceBehavior, IErrorHandler
    {
        //
        // IServiceBehavior Interface
        //
        public void AddBindingParameters(ServiceDescription serviceDescription,
           ServiceHostBase serviceHostBase, CollectionServiceEndpoint endpoints,
           BindingParameterCollection bindingParameters)
        {
        }
        public void ApplyDispatchBehavior(ServiceDescription serviceDescription,
            ServiceHostBase serviceHostBase)
        {
            foreach (ChannelDispatcher channelDispatcher in serviceHostBase.ChannelDispatchers)
{
                channelDispatcher.ErrorHandlers.Add(this);
            }
        }
        public void Validate(ServiceDescription serviceDescription, ServiceHostBase
serviceHostBase)
        {
        }

        //
        // IErrorHandler Interface
        //
        public bool HandleError(Exception e)
        {
            // Process the exception in the required way, in this case just outputting to the
console
            Console.Out.WriteLine(e);

            // Always return false to allow any other error handlers to run
            return false;
        }
        public void ProvideFault(Exception error, MessageVersion version, ref Message fault)
        {
        }
    }
```

Ošetřování různých názvů prvků odezvy SOAP

WCF očekává, že název navrácené hodnoty bude standardně ve specifickém formátu, ale služba nemusí vrátit prvek s jeho jménem v očekávaném formátu.

WCF má konvenci očekávající, že vrácená hodnota má být pojmenována v následujícím formátu: *methodName* `Result` , kde *methodName* je název operace služby. Například u služby s názvem `getQuote`očekává požadavek WCF odezvu, která má být volána: `getQuoteResult` .

Služba však může vrátit prvek s názvem, který není v souladu s tímto formátem.

Při spuštění nástroje scvutil pro generování klienta proxy, pokud WSDL uvádí jiný název, přidá rozhraní serveru proxy parametry pro instrukci WCF se jménem, na které se má hledat. Příklad:

```
[System.ServiceModel.OperationContractAttribute(Action = "", ReplyAction = "*")]
[System.ServiceModel.XmlSerializerFormatAttribute(Style =
System.ServiceModel.OperationFormatStyle.Rpc,
                                                  Use =
System.ServiceModel.OperationFormatUse.Encoded)]
[return: System.ServiceModel.MessageParameterAttribute(Name = "getQuoteReturn")]
float getQuote(string in0);
```

Pokud vytvoříte vlastní rozhraní (například přidáním metody požadavek-odezva na existující rozhraní serveru proxy), musíte zajistit, abyste přidali stejné parametry do rozhraní, pokud služba vrátí jiný název. Pokud tak neuděláte, pak nejčastějším problémem je, že volání metody služby vždy vrátí hodnotu null; je-li objekt vrácen, pak metoda vrací hodnotu null, ale pokud je vrácena numerická hodnota, jako je celé číslo, pak metoda vrací 0.

Vývoj webových služeb pomocí produktu IBM MQ

Můžete vyvíjet aplikace produktu IBM MQ pro webové služby pomocí přenosu IBM MQ pro protokol SOAP.

Informace o této úloze

Poznámka: V produktu IBM MQ 9.0je přenos IBM MQ pro protokol SOAP zamítnutý. To zahrnuje následující funkce produktu:

- Listener IBM MQ Java
- Listener IBM MQ .NET 1 a 2
- Klient IBM MQ Java Axis2
- Klient produktu IBM MQ Java (zamítnuto oznámení v produktu IBM MQ 8.0)
- Klienti IBM MQ .NET 1 a 2 (zamítnuto oznámení v produktu IBM MQ 8.0)
- IBM MQ bridge for HTTP (depreciace oznámena v IBM MQ 8.0)

Přenos IBM MQ pro SOAP poskytuje přenos JMS pro SOAP. Přenos produktu IBM MQ pro SOAP je také integrován do jiných prostředí, jako například Microsoft Windows Communication Foundation, WebSphere Application Servera CICS Transaction Server.

Další informace o transportu produktu IBM MQ pro protokol SOAP naleznete v tématu “Vyvíjení webových služeb s přenosem produktu IBM MQ pro SOAP” na stránce 1256.

Související pojmy

“Koncepty vývoje aplikací” na stránce 6
K zápisu aplikací IBM MQ můžete použít volbu procedurálních nebo objektově orientovaných jazyků. Odkazy v tomto tématu použijte pro informace o koncepcích produktu IBM MQ , které jsou užitečné pro vývojáře aplikací.

“Vyvíjení aplikací pro IBM MQ” na stránce 5
Můžete vyvíjet aplikace k odesílání a přijímání zpráv a ke správě správců front a souvisejících prostředků. IBM MQ podporuje aplikace napsané v mnoha různých jazycích a rámcích.

“Aspekty návrhu pro aplikace produktu IBM MQ” na stránce 43

Když se rozhodnete, jak vaše aplikace mohou využívat výhody platforem a prostředí, které máte k dispozici, musíte se rozhodnout, jak používat funkce nabízené produktem IBM MQ.

“Psaní procedurální aplikace pro řazení do fronty” na stránce 689
Prostřednictvím těchto informací můžete získat informace o vytváření aplikací pro řazení do front, připojování a odpojování od správce front, publikování a odběru a otevírání a zavírání objektů.

“Zápis procedurálních aplikací klienta” na stránce 874
Co potřebujete vědět, chcete-li psát klientské aplikace na serveru IBM MQ pomocí procedurálního jazyka.

“Zapisování aplikací typu publikování/odběr” na stránce 774
Začněte psát aplikace pro publikování/odběr aplikací produktu IBM MQ .

“Sestavení procedurální aplikace” na stránce 966
Aplikaci IBM MQ můžete psát v jednom z několika procedurálních jazyků a aplikaci spustit na několika různých platformách.

“Obsluha chyb procedurálních programů” na stránce 1015
Tyto informace vysvětlují chyby související s voláními MQI MQI buď při volání, nebo při doručení její zprávy do konečného cíle.

Související úlohy

“Použití ukázkových procedurálních programů produktu IBM MQ” na stránce 1034
Tyto ukázkové programy jsou napsány v procedurálních jazycích a demonstrují typická použití rozhraní MQI (Message Queue Interface). Programy produktu IBM MQ na různých platformách.

Vyvíjení webových služeb s přenosem produktu IBM MQ pro SOAP

Přenos IBM MQ pro SOAP poskytuje přenos JMS pro SOAP. Přenos produktu IBM MQ pro SOAP je také integrován do jiných prostředí, jako například Microsoft Windows Communication Foundation, WebSphere Application Servera CICS Transaction Server.

Informace o této úloze

Poznámka: V produktu IBM MQ 9.0je přenos IBM MQ pro protokol SOAP zamítnutý. To zahrnuje následující funkce produktu:

- Listener IBM MQ Java
- Listener IBM MQ .NET 1 a 2
- Klient IBM MQ Java Axis2
- Klient produktu IBM MQ Java (zamítnuto oznámení v produktu IBM MQ 8.0)
- Klienti IBM MQ .NET 1 a 2 (zamítnuto oznámení v produktu IBM MQ 8.0)
- IBM MQ bridge for HTTP (depreciace oznámena v IBM MQ 8.0)

Introduction to IBM MQ transport for SOAP

Přenos IBM MQ pro SOAP poskytuje přenos JMS pro SOAP. Odesílatel SOAP IBM MQ a modul listener poskytují prostředky pro volání webových služeb.

Modul listener protokolu SOAP produktu IBM MQ podporuje služby hostované rámcem .NET Framework 1, .NET Framework 2 a Axis 1.4. Odesilatel SOAP produktu IBM MQ podporuje klienty webových služeb spuštěné v prostředí .NET Framework 1, .NET Framework 2, Axis 1.4 a Axis2. Klienti mohou být buď server IBM MQ , nebo klientská aplikace. Přenos produktu IBM MQ pro SOAP je také integrován do jiných prostředí, jako například Microsoft Windows Communication Foundation, WebSphere Application Servera CICS Transaction Server.

Integrace do produktu Microsoft Windows Communication Foundation je součástí podpory produktu IBM MQ pro produkt .NET Framework 3.

Přenos IBM MQ pro SOAP je sada protokolů a nástrojů pro přenos zpráv SOAP pomocí produktu JMS přes IBM MQ. Je zabalen různými způsoby pro různá prostředí aplikace, jak je uvedeno v Tabulka 186 na stránce 1257.

Tabulka 186. Přenos produktu IBM MQ pro prostředí aplikace SOAP		
	Integrace s dalšími komponentami produktu IBM MQ	**Integrovaná do rámce**
Poskytnuto jako součást instalace produktu IBM MQ	.NET Rámec 1 .NET Framework 2 Axis 1.4	Windows Communication Foundation (.NET Framework 3) Axis2 (pouze klient)
Poskytnuto v jiném softwarovém balíku		WebSphere Application Server CICS Transaction Server 4.1 WebSphere ESB WebSphere Process Server for Multiplatforms

Integrace přenosu IBM MQ pro SOAP do aplikačního rámce zjednodušuje vývoj a implementaci webových služeb pro produkt IBM MQ.

S dalšími komponentami SOAP produktu IBM MQ můžete pracovat přímo s komponentami SOAP produktu IBM MQ pro vývoj a implementaci služeb. Pomocí nástrojů IBM MQ SOAP můžete konfigurovat a implementovat webové služby a klienty webové služby do produktu IBM MQ.

V integrovaných prostředích je vývoj a implementace jednodušší. Stejné nástroje použijete pro vývoj a implementaci, jako byste mohli vyvinout a implementovat webovou službu SOAP HTTP. Musíte stále konfigurovat fronty IBM MQ , kanály a správce front, které vyžadujete pomocí nástrojů produktu IBM MQ .

Můžete mixovat a porovnávat klienty a servery SOAP IBM MQ z libovolného z těchto prostředí.

Výhody

Přenos IBM MQ pro SOAP nabízí existujícím uživatelům produktu IBM MQ následující hlavní výhody:

Pomocí sítě IBM MQ můžete připojit existující webové služby.

Služby mohou být ty, které jste napsali, nebo služby poskytované jako rozhraní k jiným sbaleným softwarovým aplikacím, které jste implementovali.

Přínos pochází z použití vaší stávající sítě IBM MQ pro připojení webových služeb. Přenos produktu IBM MQ má tu výhodu, že se jedná o spravovanou a spolehlivou službu systému zpráv ve frontě.

Zápis nových aplikací nebo převod existujících aplikací tak, aby místo rozhraní produktu IBM MQ používal protokol SOAP.

Obvykle aplikace vyžadují, aby byl vyvinut specifický adaptér IBM MQ pro integraci s jinou aplikací. Adaptéry mají dvě části: část konektoru, která vkládá a získává zprávy do a z přenosu, a část adaptéru, která převádí data do a ze specifických formátů aplikace. Integrace jednotlivých párů aplikací je nová výzva.

Výhodou protokolu SOAP je standardizace protokolu SOAP pro definování aplikačních rozhraní a poté výběr transportů. Nemusíte psát specifické aplikační adaptéry a vy můžete zvolit, zda má být jako konektor použit IBM MQ nebo HTTP. Kterou dopravu si zvolíte, záleží na tom, jaké kvality služeb a připojení požadujete.

Pro existující použití protokolu SOAP přes HTTP je přínos transportu produktu IBM MQ pro protokol SOAP z použití spravované a spolehlivé asynchronní přenosu. Výhody jsou dvojí:

Skutečně asynchronní programovací model pro dostupnost a výkon.
Pomocí asynchronního rozhraní klienta nemusí být klient a aplikace služeb k dispozici ve stejnou dobu. Požadavky odeslané klientem budou uloženy až do doby, kdy bude služba k dispozici pro zpracování.

Připravená sestavená spravovaná síť, která je navržena tak, aby byla spolehlivá a dostupná.

Vyberete-li volbu IBM MQ jako transport, budete využívat výhod používání spravované sítě, která poskytuje spolehlivý systém zpráv.

Naproti tomu přenosy, jako jsou HTTP a FTP přes TCP/IP, jsou nespravované. Nespravovaná síť je ideální pro nepředvídatelné připojení: je zde méně úloh správy.

Souhrn

Přenos produktu IBM MQ pro SOAP poskytuje následující komponenty:

- Vazba přenosu SOAP/JMS se používá v dokumentech WSDL k vazbě služby SOAP s přenosem JMS . Implementace IBM MQ vazby SOAP/JMS používá identifikátor URI, který přijímá některou z následujících dvou formátů:

  Transport produktu IBM MQ pro protokol SOAP

  ```
  jms:/queue? &Name=Value&Name=Value...
  ```

  Formát spoje IBM MQ pro doporučení kandidáta W3C

  ```
  jms:queue: qName ?connectionFactory=connectQueueManager
  (qMgrName)&Name=Value&Name=Value...
  ```

- Mapování zprávy SOAP na zprávu IBM MQ .
- Dva moduly listener SOAP IBM MQ pro příjem požadavků SOAP, jeden pro produkt Java a druhý pro rámec .NET Framework 1 nebo .NET Framework 2. Moduly listener používají .NET nebo Axis 1.4 ke zpracování požadavku SOAP.
- Dva odesilatelé SOAP IBM MQ , aby vytvořili požadavky SOAP IBM MQ . Klienti webových služeb se registrují s odesilatelem pro zpracování požadavků SOAP `jms:` .
- Integrace s produktem Windows Communication Foundation (WCF), někdy známá jako .NET 3, pro odesílání a příjem IBM MQ přenosu zpráv SOAP.
- Integrace klienta s Axis2, někdy označovaná jako rozhraní JAX-WS, pro odesílání zpráv IBM MQ Transport pro SOAP nebo W3C SOAP JMS .
- Příkaz **amqwdeployWMQService**, který vytváří komponenty pro vývoj a běhové komponenty a skripty k implementaci webové služby pomocí přenosu protokolu SOAP produktu IBM MQ .
- Ukázka klienta a kódu služeb Java a .NET .
- Skript pro nastavení cesty ke třídě a ostatní skripty obslužných programů.

V integrovaných prostředích jsou odesilatel a modul listener integrováni do každého prostředí, stejně jako rozšíření nástrojů pro vývoj a implementaci.

Integrace SOAP a IBM MQ

Přenos produktu IBM MQ pro protokol SOAP rozšiřuje nástroje SOAP a nástroje webových služeb a běhové prostředí s produktem IBM MQ jako alternativní transport k protokolu HTTP pro protokol SOAP. Existující webové služby nemusíte upravovat, abyste mohli používat přenos IBM MQ pro protokol SOAP jako přenos. Přenos používá vlastní formát identifikátoru URI pro SOAP/JMS. Formát identifikátoru URI W3C pro SOAP/JMS je podporován omezeným způsobem klienty Axis2 .

Dodatečný řádek kódu musí být přidán do klientů v prostředích .NET Framework 1, .NET Framework 2 a Axis 1.4 . V klientech Axis 2 a Windows Communication Foundation (WCF) není vyžadován žádný další kód. Modul listener SOAP IBM MQ spouští služby v prostředí .NET Framework 1, .NET Framework 2 a Axis 1.4 . Přenos produktu IBM MQ pro SOAP je integrován do některých dalších prostředí aplikačního serveru, včetně WCF, CICSa WebSphere Application Server.

Co je SOAP?

Protokol SOAP[11]popisuje standardizovaný formát zpráv a protokolů interakce, které aplikace používají k výměně požadavků, odpovědí a datagramů. Protokol SOAP je nezávislý na přenosu, který se používá

[11] Historicky se zkratka stala pro protokol SOAP (Simple Object Access Protocol).

k přenosu zpráv, a prostředí aplikace, které odesílá a přijímá zprávy. W3C definuje protokol SOAP 1.2 (stručný popis):

SOAP 1.2 poskytuje definici informací na bázi XML, které lze použít pro výměnu strukturovaných a typovaných informací mezi rovnocennými partnery v decentralizovaném, distribuovaném prostředí.[12].

Chcete-li použít SOAP, musí být svázán s přenosem, jako je HTTP, e-mail nebo IBM MQ.

Rámec vazeb protokolu SOAP je sada pravidel pro přenos zprávy SOAP nad jiným protokolem, jako je například HTTP. SOAP 1.2 Část 2: Adjuncts (Second Edition) popisuje vazbu SOAP HTTP.

Doporučení pro doporučení W3C , 4. června 2009, SOAP over Java Message Service 1.0, popisuje doporučení pro vazbu SOAP JMS . As JMS is an API specification, and not a transport protocol, the JMS SOAP recommendation does not describe the wire-format of SOAP JMS messages. Popisuje protokoly interakce SOAP a vazbu rozhraní API produktu JMS . V důsledku toho při použití doporučení SOAP JMS musíte i nadále používat stejnou implementaci produktu JMS pro klienta SOAP a server SOAP. To umožňuje spuštění aplikace SOAP JMS na libovolné implementaci produktu JMS. Implementace produktu JMS může být připojena k aplikačnímu serveru J2EE , pokud server i implementace produktu JMS vyhovují specifikaci JCA. IBM MQ Produkt JMS odpovídá specifikaci JCA a lze jej zapojit do odpovídajícího aplikačního serveru.

IBM MQ transport pro vazbu SOAP je stejně jako navrhovaný standard W3C , ale není to stejný. Jeho použití je popsáno v tématu Nastavení protokolu SOAPMQRFH2. Na rozdíl od doporučení kandidáta W3C není vazba SOAP formálně uvedena. Efektivně je to vazba HTTP a adresa služby má formu `jms:/queue?name=value&name=value...`, spíše než `http://authority/path?query#fragment`. `jms:` není oficiálně registrovaný schéma identifikátoru URI IANA.

Co je webová služba?

Protokol SOAP umožňuje, aby programy napsané v různých jazycích, které jsou spuštěny na různých platformách, komunikovaly pomocí různých přenosových protokolů. SOAP je specifikace protokolu. Webová služba je aplikace, která poskytuje službu prostřednictvím rozhraní SOAP, ke kterému lze přistupovat prostřednictvím internetových protokolů.

Důležitým cílem protokolu SOAP je poskytovat služby, které klienti mohou používat snadno. Jakmile jste navrhli, aby klient používal službu, můžete naprogramovat volání k vyvolání služby bez odkazu na externí dokumentaci. Rozhraní služby jsou popsána v XML v dokumentu WSDL. Dotaz, `http://authority/path?wsdl`, vrací popis WSDL služby SOAP.

Tip: Když implementujete webovou službu pro použití produktu IBM MQ, implementujte také službu do protokolu HTTP tak, aby standardní dotaz WSDL pracoval.

Vývoj webových služeb

Webové služby mají klienta a část služby. Služba se zapíše jako první, buď začíná od popisu rozhraní ve WSDL, nebo podle pravidel pro zápis třídy služeb. Sady nástrojů webových služeb mají obslužné programy pro generování kódu WSDL z definice rozhraní třídy; například **`java2wsdl`** nebo **`disco`**. Mají také nástroje pro generování nebo třídy_třídy z popisů rozhraní WSDL, například **`wsdl2java`**, **`wsimport`**nebo **`wsdl`**. První je známý jako vývoj zdola nahoru, a druhý shora dolů.

Příkaz **`amqwdeployWMQService`** v IBM MQ přenosu pro SOAP používá tyto nástroje ke generování souborů WSDL, stubů klienta a proxy klienta.

Webové služby jsou obvykle napsány pomocí integrovaného vývojového prostředí zaměřeného na konkrétní prostředí aplikačního serveru:

Eclipse IDE pro vývojáře Java EE
Vytvoří webové služby pro osu 2. Podporuje JAX-RPC a JAX-WS

[12] W3C: SOAP 1.2 -část 0

Rational Application Developer 7.5
Vytvoří webové služby pro produkt WebSphere Application Server V7 a předchozí verze a také pro osu. Podporuje JAX-RPC a JAX-WS.

WebSphere Integration Developer 6.2
Vytvoří webové služby pro produkt WebSphere Process Server a WebSphere ESB. Podporuje JAX-RPC a JAX-WS.

Visual Studio 2012
Vytvoří webové služby pro produkt .NET Framework 3.5 a starší (Windows Communication Foundation)

Můžete použít kterýkoli z těchto nástrojů v kombinaci s přenosem IBM MQ pro SOAP. Jakmile jste vyvinuli službu pro použití s HTTP, použijte nástroj **amqwdeployWMQService** k implementaci služeb pro použití produktu IBM MQ jako přenosu. Můžete zapsat nového klienta pomocí výstupu z nástroje nebo upravit existující klienty, aby mohli používat přenos IBM MQ pro SOAP.

Pokud je přenos IBM MQ pro SOAP integrován do prostředí aplikace, pak nemusíte používat nástroj **amqwdeployWMQService** nebo můžete upravit kód klienta. Vrstva SOAP klienta směruje požadavky klientů, které mají identifikátor URI s předponou `jms:` , do transportu produktu IBM MQ pro protokol SOAP. Vrstva SOAP serveru volá přenos IBM MQ , aby protokol SOAP čekal na požadavky SOAP `jms:` a vrátil odpovědi na přenos IBM MQ pro SOAP.

Služby produktu .NET jsou obvykle vytvářeny za použití anotací webových služeb v kódu a služeb produktu Java shora dolů pomocí definic rozhraní WSDL. Rozdíl v přístupech se zmenšuje, protože produkt Java Standard Edition 6 podporuje rozhraní JAX-WS 2.0a používá anotace ke kvalifikaci definice rozhraní služby. Nyní je snadné vyvinout služby Java zdola nahoru jako shora dolů. Který přístup zvolíte je otázka metody vývoje.

Klient webových služeb je zapsán po službě pomocí definice služby WSDL a generovaných stubů klienta a serverů proxy. V některých aplikacích není definice služby známá, když je klient zapsán. Klient načte WSDL služby a vytvoří požadavky na službu dynamicky. Obecněji je známá definice služby, ale adresa, na kterou je služba implementována, není známa. Sada nástrojů webové služby generuje rozhraní pro klienta, který se má použít k provádění požadavků na službu. Klient poskytuje adresu služby, je-li požadována. Ve třetím případě obsahuje WSDL všechny informace, které klient potřebuje. WSDL obsahuje jak rozhraní, tak adresu služby. Kód generovaný sadou nástrojů webové služby má všechny informace, které klient potřebuje k provádění požadavků na službu.

Pro SOAP je možné použít kterýkoli z těchto tří stylů s přenosem IBM MQ .

Aplikační prostředí webových služeb

Sady nástrojů webových služeb vyžadují mapování z definice WSDL služby na proudy bajtů, které jsou přeneseny v požadavcích a odezvách SOAP. Bajtový proud je definován specifikací protokolu SOAP a je obsažen v obálce SOAP. Obálka SOAP je zobrazena v části Obrázek 163 na stránce 1260.

```
<?xml version='1.0'?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Header> <!-- optional -->

<!-- headers... -->

</soap:Header>
<soap:Body>
<!-- payload or fault message -->

</soap:Body>
</soap:Envelope>
```

Obrázek 163. obálka SOAP

Mapování z obálky SOAP na vazbu jazyka a zpět je standardizovaný a částečně autorizovaný. Mapování je základní pro architekturu produktu .NET a je poskytováno jako součást běhového prostředí CLR (Common Language Runtime). Mapování je standardizováno v produktu Java specifikací JAX. Protože jsou mapování

Java standardizována, jsou klienti a služby webových služeb produktu Java přenositelné mezi různými aplikačními prostředími založená na produktu Java. Rozhraní JAX-RPC (někdy nazývané JAX-WS 1.0) je mapování, které se nejvíce používá dnes. Je podporován Axis 1.4. JAX-WS (někdy označovaný jako JAX-WS 2.0) je výrazně vylepšený standard a pravděpodobně bude rychle nahrazovat JAX-RPC. JAX-WS je podporován Axis 2.0. IBM MQ 7.0.1 nepodporuje rozhraní JAX-WS a Axis 2.

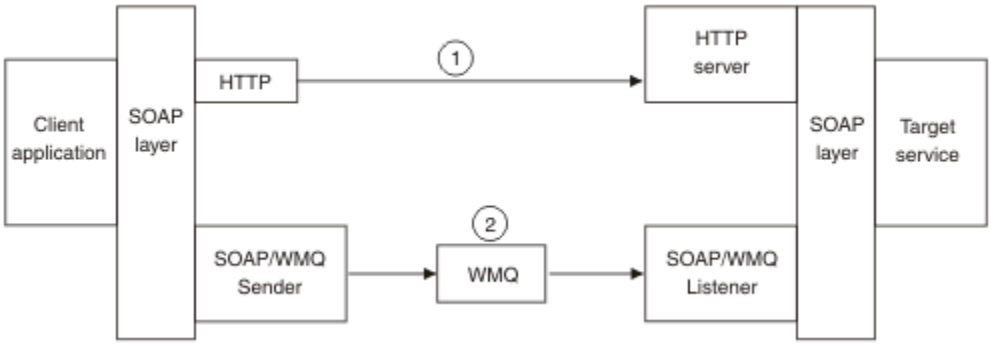
IBM MQ přenos pro SOAP neupravuje obsah obálky SOAP a obsah neovlivní přenos. Vazby jazyka mají vliv na přenos IBM MQ pro protokol SOAP. IBM MQ 7.0.1 podporuje .NET Framework 1, .NET Framework 2 a Axis 1.4 za použití kódu a obslužných programů dodaných s přenosem IBM MQ pro protokol SOAP. Podpora pro přenos WebSphere pro SOAP v produktu .NET Framework 3 a 3.5 je implementována pomocí vlastního kanálu IBM MQ pro produkt Windows Communication Foundation.

Další vývojová prostředí SOAP a běhová prostředí mohou poskytovat podporu transportu produktu IBM MQ pro protokol SOAP a podporovat různé jazyky. Například webové služby spuštěné v produktu CICS podporují jazyky, jako jsou COBOL a PL/1.

Poznámka: Použité mapování nečiní žádný rozdíl v interoperabilitě webových služeb. Můžete mixovat a porovnávat klienty a služby napsané pomocí mapování .NET, JAX-RPC a JAX-WS.

Co je přenos IBM MQ pro SOAP?

Transport produktu IBM MQ pro SOAP je vazba SOAP a sada nástrojů webových služeb. Společně umožňují výměnu zpráv SOAP pomocí produktu IBM MQ spíše než HTTP. Obrázek 164 na stránce 1261 shows IBM MQ as an alternative to HTTP as a SOAP transport.

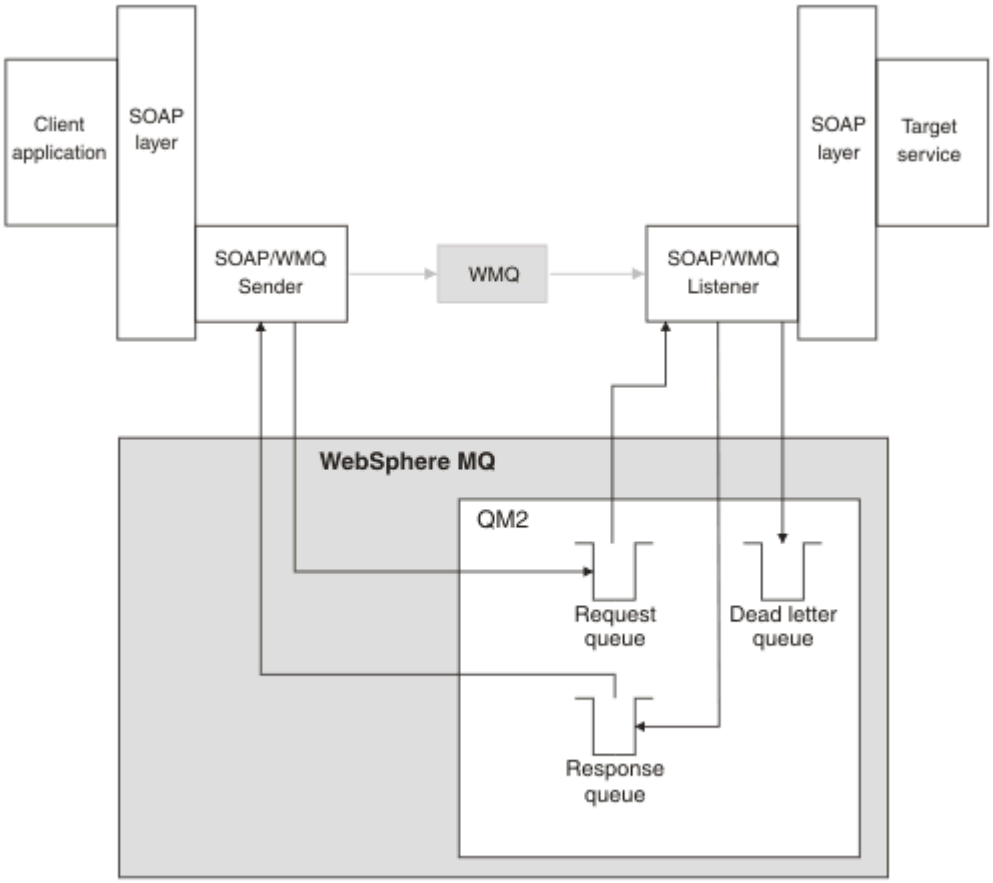


Obrázek 164. Přehled přenosu produktu IBM MQ pro protokol SOAP

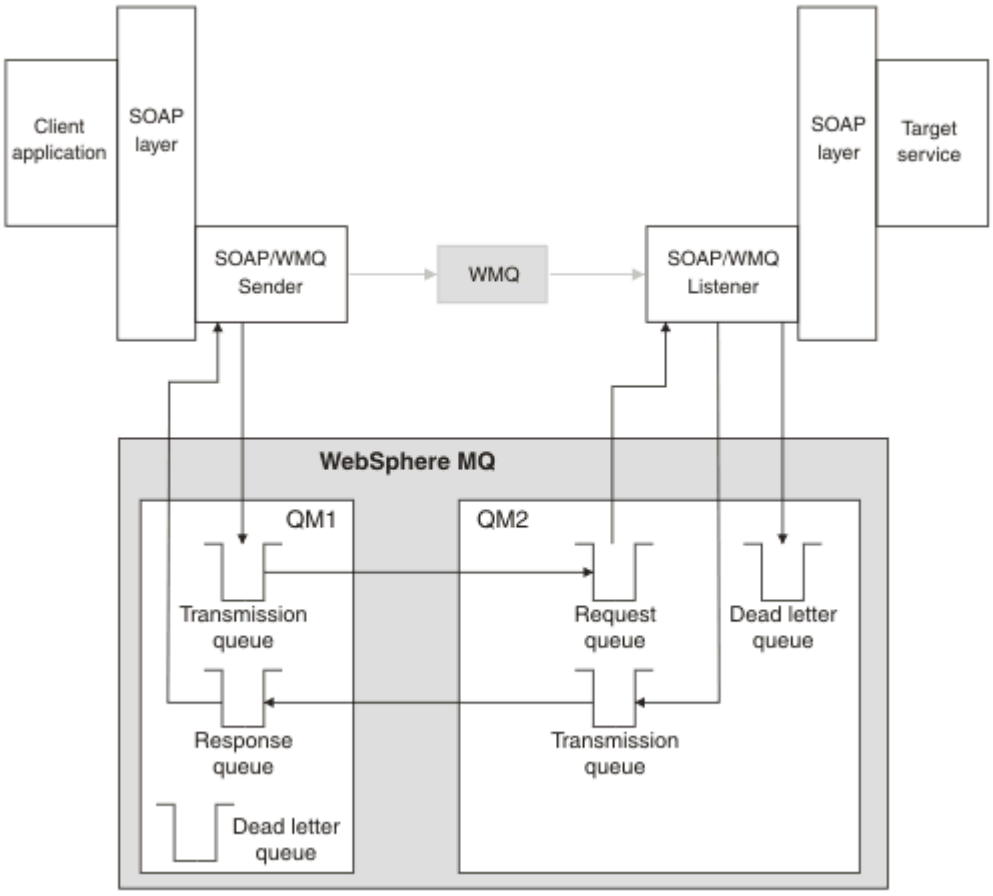
Protokol SOAP prostřednictvím protokolu HTTP se v diagramu zobrazuje jako (1). Vrstva SOAP klienta převádí požadavek do zprávy SOAP a komponenta HTTP posílá přes protokol TCP/IP. Komponenta serveru HTTP naslouchá požadavkům HTTP, obvykle na portu TCP/IP 80. Je-li požadavek pro službu SOAP, volá komponenta serveru HTTP vrstvu SOAP, aby převedla požadavek SOAP na volání metody. Poté vrátí odezvu.

SOAP přes IBM MQ je zobrazeno jako (2). Klientská aplikace registruje komponentu odesílatele SOAP IBM MQ jako obslužnou rutinu pro protokol `jms:` se vrstvou SOAP. Vrstva SOAP předává zprávy SOAP adresované `jms:` odesilateli SOAP IBM MQ adresovaným zprávám SOAP. Odesílatel používá identifikátor URI ve zprávě k umístění zprávy do fronty požadavků s použitím požadovaných kvalit služby. The corresponding IBM MQ SOAP listener waits for messages on its request queue and calls the SOAP layer to process requests and return responses.

Odesílatel SOAP a modul listener jsou normální IBM MQ programy. Mohou být připojeny ke stejnému správci front jako v produktu Obrázek 165 na stránce 1262nebo jsou připojeny k různým správcům front; viz Obrázek 166 na stránce 1263. Klient může být připojen pomocí připojení klienta.



Obrázek 165. Fronty použité SOAP/IBM MQ (jednotlivý správce front)



Obrázek 166. Fronty použité produktem SOAP/IBM MQ (samostatné správce front)

Doporučené doporučení W3C pro vazbu protokolu SOAP s produktem JMS.

Doporučení kandidáta W3C definuje protokol SOAP nad vazbou služby JMS; Protokol SOAP nad Java Message Service 1.0. Také užitečné pro jeho příklady je Schéma identifikátoru URI pro službu Java(tm) Message Service 1.0[13].

Některé struktury aplikací, jako např. WebSphere Application Server v7, mají podporu pro doporučení kandidáta W3C . Odeslat požadavky SOAP formátované pomocí identifikátoru URI kompatibilního s doporučujícím doporučením W3C s použitím klienta Axis2 ; viz W3C SOAP over JMS URI pro klienta IBM MQ Axis 2. Klient Axis2 odešle požadavek SOAP formátovaný buď s přenosem W3C , nebo IBM MQ pro SOAP na základě URI v požadavku SOAP.

Podpora klienta Axis2 pro doporučení W3C je představena v opravné sadě 7.0.1.3 . Podpora pro jiné klienty a pro moduly listener SOAP poskytované produktem IBM MQ není poskytována.

Související pojmy

Implementace přenosu WebSphere pro protokol SOAP na systému .NET Framework 1, .NET 2 a Axis 1.4
You might want to write your own IBM MQ SOAP sender and listener. Jako vodítko použijte implementaci přenosu IBM MQ pro SOAP na .NET Framework 1, .NET Framework 2 a Axis 1.4 jako vodítko.

Přenos produktu IBM MQ pro spolehlivý systém zpráv SOAP a webových služeb
Spolehlivý systém zpráv webových služeb je protokol, který spolehlivě vyměňuje požadavky webových služeb a odpovědi na nespolehlivé připojení. Je nejvhodnější pro řešení problémů přerušení spojení s krátkou životností.

[13] Vyhledejte *Schéma identifikátoru URI pro rozhraní JMS* v odkazech specifikace W3C pro nejnovější koncept.

Implementace přenosu WebSphere pro protokol SOAP na systému .NET Framework 1, .NET 2 a Axis 1.4

You might want to write your own IBM MQ SOAP sender and listener. Jako vodítko použijte implementaci přenosu IBM MQ pro SOAP na .NET Framework 1, .NET Framework 2 a Axis 1.4 jako vodítko.

1. Klientský program používá vhodný rámec webových služeb stejným způsobem, jaký by měl pro přenos HTTP. Musí také zaregistrovat předponu `jms:` . Předpona se zaregistruje buď pomocí metody `com.ibm.mq.soap.Register.extension()` Java , nebo pomocí metody `IBM.WMQSOAP.Register.Extension()` CLR.
2. Rámec Axis 1.4 nebo .NET Framework 1 nebo 2 shrne volání do zprávy požadavku SOAP přesně jako pro SOAP/HTTP.
3. Služba IBM MQ je identifikována identifikátorem URI s předponou `jms:`. Když rámec identifikuje identifikátor URI produktu `jms:` , zavolá kód odesilatele transportu produktu IBM MQ , `com.ibm.mq.soap.transport.jms.WMQSender` (pro osu 1.4) nebo `IBM.WMQSOAP.MQWebRequest` (pro .NET1 a 2). Pokud rámec narazí na identifikátor URI s předponou `http:` , zavolá odesilatele protokolu SOAP přes HTTP.
4. Zpráva SOAP je přenášena odesilatelem SOAP IBM MQ pomocí fronty požadavků. **`SimpleJavaListener`** (pro Java) nebo **`amqwSOAPNETListener`** (pro .NET) přijme zprávu požadavku.

   Moduly listener protokolu SOAP produktu IBM MQ jsou samostatné procesy a jsou s podporou podprocesů s přizpůsobným počtem podprocesů.
5. Modul listener protokolu SOAP produktu IBM MQ načte příchozí požadavek SOAP a předá jej do příslušné infrastruktury webové služby.
6. Infrastruktura webové služby analyzuje zprávu požadavku SOAP a vyvolává službu. Procedura je stejná jako u zprávy, která dorazila na přenos HTTP.
7. Infrastruktura formátuje odezvu na zprávu odpovědi SOAP a vrací ji do listeneru SOAP IBM MQ .
8. Modul listener umístí zprávu na frontu odpovědí a zpráva se přenese do odesilatele SOAP IBM MQ . Odesílatel ji předá do infrastruktury webové služby klienta.
9. Infrastruktura klienta zanalyzuje zprávu odezvy SOAP a předá výsledek zpět aplikaci klienta.

Každý kontext aplikace je obsluhován samostatnou frontou požadavků produktu IBM MQ .

Kontext aplikace je řízen v Axis 1.4 tím, že se ujistí, že modul listener a služba IBM MQ SOAP se provedou v příslušném adresáři. Osa 1.4 nastavuje správnou proměnnou CLASSPATH pro daný adresář.

Kontext aplikace je řízen produktem .NET modulem listener protokolu SOAP produktu IBM MQ , který spouští službu v kontextu vytvořeném voláním příkazu `ApplicationHost.CreateApplicationHost`. Volání určuje cílový adresář provedení. Každá služba poté pracuje v adresáři, ve kterém byla implementována.

`amqwdeployWMQService` generuje fronty požadavků a požadavků. Vygeneruje také infrastrukturu potřebnou pro práci s frontami a implementaci služeb na Axis 1.4.

Související pojmy

Integrace SOAP a IBM MQ

Přenos produktu IBM MQ pro spolehlivý systém zpráv SOAP a webových služeb
Spolehlivý systém zpráv webových služeb je protokol, který spolehlivě vyměňuje požadavky webových služeb a odpovědi na nespolehlivé připojení. Je nejvhodnější pro řešení problémů přerušení spojení s krátkou životností.

Přenos produktu IBM MQ pro spolehlivý systém zpráv SOAP a webových služeb

Spolehlivý systém zpráv webových služeb je protokol, který spolehlivě vyměňuje požadavky webových služeb a odpovědi na nespolehlivé připojení. Je nejvhodnější pro řešení problémů přerušení spojení s krátkou životností.

IBM MQ for SOAP využívá pro posílání zpráv SOAP správu IBM MQ spravované a spolehlivé sítě. Přenosy jako HTTP a FTP jsou nespravované. Nespravované sítě jsou ideální pro nepředvídané připojení, kde potíže a náklady na správu připojení převažují nad výhodami neztrácející požadavky a odezvy.

Chcete-li překonat problém ztráty souborů při přerušení spojení v nespravovaných sítích, služby jako spravované FTP vytvářejí vrstvu správy na vrcholu FTP. Vrstva správy přebírá zátěž kontroly, že soubory byly úspěšně přeneseny od uživatelů, a v případě potřeby opakovaně přenáší chybějící soubory. Chcete-li používat spravovaný FTP, musíte mít nainstalovaný software pro správu na obou koncích připojení.

Spolehlivý systém zpráv webových služeb (WSRM) používá odlišný přístup k řešení problému nespolehlivých připojení. Jeho cílem je spolehlivě přenášet požadavky a odpovědi webových služeb, aniž by oba konce připojení museli používat stejný software. Veškerý software, který implementuje protokol spolehlivého systému zpráv webových služeb, může spolehlivě vyměňovat zprávy s jiným softwarem.

Dojde-li k selhání připojení, musí odesilatel a příjemce zachovat kontext přenosu zpráv WSRM s použitím generovaného identifikátoru URI jako klíče. Odesílatel a příjemce se neustále pokouší o vytvoření nového připojení. Je-li úspěšně ustanoveno nové připojení, přenos se dokončí. Specifikace WSRM neuvádí, jak je kontext zachován, nebo když se pokusíte o nové připojení.

Můžete se rozhodnout, že se zajímají pouze výpadky s krátkou životností. V případě delších výpadků můžete být připraveni zrušit přenosy, které nebylo možné po určité době restartovat. Podobně můžete být připraveni vyřadit přenosy v případě, že selže klient nebo služba. Ponechání uživatele zodpovědného za zajišťování přenosů, klade méně požadavků na správu koordinace klienta a služby.

Pokud jsou výpadky sítě typu long-žili déle než 30 minut nebo pokud dojde k selhání klienta nebo serveru, je zde zvýšená pravděpodobnost, že některá připojení nebudou nikdy znovu zavedena. Nyní již nelze spoléhat na to, že služba WSRM bude automaticky obnovovat přenos zpráv v nespravovaném způsobu. Je třeba zvážit správu nezdařených připojení WSRM, což znamená vývoj softwaru pro správu sítě klientů a služeb.

Použití WSRM k překonání krátkých výpadků by mohlo výrazně snížit práci se ztrátou zpráv v mobilní síti. Pokud nebudete muset zajistit doručování zpráv, mohou výhody snížení ztráty zpráv ospravedlnit další náklady na vývoj implementace WSRM.

Protokol SOAP prostřednictvím produktu JMS poskytuje zajištěné doručování zpráv a zabývá se delší dobou trvání výpadků klienta, serveru a sítě. Pokud hledáte spolehlivější kvalitu služby pro protokol SOAP než protokol HTTP, které řešení vyberete: IBM MQ transport pro SOAP nebo WSRM? Odpověď závisí na mnoha faktorech. Některé z faktorů, které je třeba zvážit, jsou uvedeny níže:

1. Určuje, zda je nespolehlivost způsobena selháním připojení.
2. Jak dlouho mají nemilovaná selhání spojení.
3. Pokud můžete spravovat jak straně klienta, tak i straně serveru připojení.
4. Pokud je uživatel nebo administrátor v konečném důsledku odpovědný za doručování zpráv.

Související pojmy

Integrace SOAP a IBM MQ

Implementace přenosu WebSphere pro protokol SOAP na systému .NET Framework 1, .NET 2 a Axis 1.4
You might want to write your own IBM MQ SOAP sender and listener. Jako vodítko použijte implementaci přenosu IBM MQ pro SOAP na .NET Framework 1, .NET Framework 2 a Axis 1.4 jako vodítko.

Instalace a ověření webových služeb produktu IBM MQ

Pomocí pokynů v těchto tématech nainstalujte a ověřte přenos IBM MQ pro SOAP.

Instalace produktu IBM MQ Web Transport pro SOAP

Tyto pokyny slouží k instalaci webového transportu produktu IBM MQ pro protokol SOAP. Instalace vytváří nástroje pro spouštění klientů a služeb webových služeb pomocí produktu IBM MQ jako přenosu SOAP. Nástroje se používají v prostředí .NET Framework 1, .NET 2, Axis 1.4nebo Axis2 SOAP.

Než začnete

Zkontrolujte předpokládané produkty na adrese Systémové požadavky pro IBM MQ. Proces instalace nekontroluje přítomnost a dostupnost předem vyžadovaného softwaru. Musíte ověřit, zda jsou předpoklady nainstalovány.

Produkt IBM MQ poskytuje kopii běhového prostředí Axis 1.4 . Použijte tuto verzi s produktem IBM MQ namísto jiného, který byste mohli instalovat. Produkt IBM neposkytuje technickou podporu pro osu Apache . Kontaktujte Softwarovou nadaci Apache , pokud s ní máte technické problémy.

Chcete-li spustit webové služby v prostředí .NET Framework 3 SOAP, produkt IBM MQ používá Windows Communication Foundation. Vlastní kanál produktu IBM MQ pro produkt Windows Communication Foundation spouští klienty a služby webových služeb pomocí produktu IBM MQ jako transportu pro zprávy SOAP.

Informace o této úloze

Můžete nainstalovat webový transport IBM MQ pro SOAP jako IBM MQ MQI client nebo serverová aplikace. Pokud jste již produkt IBM MQ nainstalovali jako klienta nebo server na vašem počítači, zkontrolujte, zda jste nainstalovali uvedené komponenty.

MQ_INSTALLATION_PATH představuje adresář vysoké úrovně, ve kterém je nainstalován produkt IBM MQ .

Proveďte následující kroky instalace.

Postup

1. Vyberte komponentu produktu `Java and .Net Messaging and web services` pro instalaci.
2. V systémech Solaris a HP-UXvyberte pro instalaci komponentu produktu `Java Runtime environment` .
3. Vyberte sadu vývojových nástrojů pro instalaci.
4. Nainstalujte a ověřte IBM MQ , jak je popsáno v Stručné úvodní části pro vaši platformu.
5. Okopírujte běhové prostředí Apache Axis 1.4 , `axis.jar` z adresáře `prereqs/axis` na instalačním médiu produktu IBM MQ . Zkopírujte jej do instalačního adresáře popsaného v části Tabulka 187 na stránce 1267, Tabulka 188 na stránce 1267nebo Tabulka 189 na stránce 1267.

   Windows

   ```
   Copy D:\PreReqs\axis\axis.jar MQ_INSTALLATION_PATH\java\lib\soap
   ```

   AIX

   ```
   cp -f PreReqs/axis/axis.jar MQ_INSTALLATION_PATH/java/lib/soap/axis.jar
   chown mqm:mqm MQ_INSTALLATION_PATH/java/lib/soap/axis.jar
   chmod 444 MQ_INSTALLATION_PATH/java/lib/soap/axis.jar
   ```

   Instalační adresáře produktů HP-UX, Solarisa Linux (všechny platformy)

   ```
   cp -f PreReqs/axis/axis.jar MQ_INSTALLATION_PATH/java/lib/soap/axis.jar
   chown mqm:mqm MQ_INSTALLATION_PATH/java/lib/soap/axis.jar
   chmod 444 MQ_INSTALLATION_PATH/java/lib/soap/axis.jar
   ```

6. V systému Windows 2003spusťte **Aspnet_regiis.exe** a aktualizujte mapy skriptů tak, aby ukazovaly na verzi běhového prostředí obecného jazyka, kterou používáte.

   Hledejte obslužný program **Aspnet_regiis.exe** v produktu %SystemRoot%Microsoft.NET/Framework/*version-number*.
7. Nastavte proměnnou prostředí WMQSOAP_HOMEtak, aby ukazovala na instalační adresář produktu IBM MQ .

Výsledky

Tabulka 187. Instalační adresáře produktu Windows	
Umístění	**Obsah**
MQ_INSTALLATION_PATH \programs\bin	Binární, příkaz, DLL a spustitelné soubory
MQ_INSTALLATION_PATH \programs\java\lib	`.jar files`
MQ_INSTALLATION_PATH \programs\java\lib\soap	Protokol SOAP `.jar files`
MQ_INSTALLATION_PATH \programs\soap\samples	Vzorky a IVT

Tabulka 188. Instalační adresáře produktu AIX	
Umístění	**Obsah**
MQ_INSTALLATION_PATH/bin	Skripty shell
MQ_INSTALLATION_PATH/java/lib	`.jar files`
MQ_INSTALLATION_PATH/java/lib/soap	Soubory `axis.jar` a další soubory `.jar` JAX-RPC
MQ_INSTALLATION_PATH/samp/soap	Vzorky a IVT

Tabulka 189. Instalační adresáře produktů HP-UX, Solarisa Linux (všechny platformy)	
Umístění	**Obsah**
MQ_INSTALLATION_PATH/bin	Skripty shell
MQ_INSTALLATION_PATH/java/lib	`.jar files`
MQ_INSTALLATION_PATH/java/lib/soap	Soubory `axis.jar` a další soubory `.jar` JAX-RPC
MQ_INSTALLATION_PATH/samp/soap	Vzorky a IVT

Jak pokračovat dále

1. Pouze pro .NET , musíte registrovat přenos IBM MQ pro soubory SOAP s globální mezipamětí sestavení. Je-li produkt .NET již nainstalován při instalaci produktu IBM MQ, bude registrace provedena automaticky při instalaci. Pokud nainstalujete .NET po IBM MQ, registrace se provede automaticky, když se IVT poprvé spustí.

   Produkt **amqiregisterdotnet.cmd** můžete spustit k provedení registrace montážních celků .NET . Můžete také spustit příkaz **amqiregisterdotnet.cmd** k vynucení opětovné registrace v libovolné fázi. Po provedení této registrace přežije restart systému a následná opětovná registrace není obvykle nutná.
2. Spusťte test ověření instalace, jak je popsáno v tématu “Ověření transportu produktu IBM MQ pro protokol SOAP” na stránce 1267.
3. Hodláte-li vyvíjet klienta Axis2 , musíte stáhnout Axis2 1.4.1 z Apache; viz “Vyvíjení klienta JAX-WS pro produkt WebSphere transport pro protokol SOAP s použitím prostředí Eclipse” na stránce 1285.

Ověření transportu produktu IBM MQ pro protokol SOAP

Ověřte přenos IBM MQ pro protokol SOAP pomocí příkazu **runivt** . Příkaz spustí celou řadu demonstračních aplikací a zajistí, aby prostředí bylo po instalaci správně nastaveno. Všimněte si, že webová služba části přenosu pro SOAP je zamítnuta, a pokud jste novým uživatelem, neměli byste ji používat.

Než začnete

Než spustíte příkaz **runivt** , ujistěte se, že máte následující běhová prostředí:

- Chcete-li spustit pouze na ose Axis, musíte mít ve svém systému k dispozici sadu Java SDK (within SOE). Musíte také zahrnout umístění příkazů `java.exe` a `javac.exe` v systémové proměnné prostředí **PATH** .
- To run a test on .NET only (supported only on Windows): you must have both a Java SDK and the .NET compilers and tools on your system. Chcete-li tak učinit, vstupte buď do příkazového řádku produktu Visual Studio, nebo do příkazového řádku Microsoft Windows SDK, a potom přidejte umístění souborů `java.exe` a `javac.exe` do proměnné prostředí **PATH** .
- Chcete-li spustit všechny dostupné testy: pro platformy Windows , musí být prostředí nakonfigurováno způsobem popsaným v testovacím běhu produktu .NET . Na platformách UNIX and Linux musí být prostředí konfigurováno tak, jak je popsáno v testovacím běhu pouze Axis.

Informace o této úloze

Místo spouštění testu verifikace na produktu .NET i na ose Axis možná budete chtít spustit test pouze na ose Axis nebo pouze na .NET.

Pokud se setkáte s problémy s testy a chcete znovu začít:

1. Zastavte správce front `WMQSOAP.DEMO.QM` pomocí volby `immediate` .
2. Zastavte modul listener, který byl spuštěn v jiném okně.
3. Odstraňte správce front.
4. Odstraňte adresář dočasných ukázek, který jste vytvořili a začněte znovu.

Na platformách UNIX and Linux je nutné spustit příkaz pomocí systémové relace X Windows .

Příkaz **runivt** mění obsah adresáře `soap/samples` . Chcete-li uchovat obraz instalace nezměněný, zkopírujte adresář ukázek do dočasného umístění a spusťte verifikační test z dočasného umístění.

Verifikaci instalace můžete spustit tolikrát, kolikrát chcete.

Proveďte následující kroky, abyste ověřili instalaci přenosu IBM MQ pro SOAP na .NET Framework 1, .NET Framework 2 a Axis 1.4:

Postup

1. Zkopírujte adresářový strom `./tools/soap/samples` do dočasného umístění.
2. Spusťte příkazové okno s dočasným adresářem jako aktuální adresář.
3. Použijte příkaz **runivt** ke spuštění testu instalace. Skript `runivt` zkompiluje testovací třídu, ukázkového klienta a služby před implementací a spuštěním těchto tříd. Pro testovací třídu, ukázkový klient a služby ke spuštění dokončete kroky instalace uvedené v tématu Instalace produktu WebSphere(r) MQ Web Transport for SOAP a ujistěte se, že příkazový řádek použitý ke spuštění příkazu `runivt` má nastavenou běhovou sadu prostředí. Ke spuštění příkazu **runivt** použijte libovolnou z následujících metod:
   - Spustit test pouze na ose Axis: `runivt Axis`.
   - Spustit test pouze na .NET (podporováno pouze na Windows): `runivt DotNet`.
   - Spusťte všechny dostupné testy: `runivt`.

   Další informace o syntaxi a parametrech příkazu `runivt` naleznete v příručce **runivt**: IBM MQ transport for SOAP installation verification test. Testy, které můžete spustit, jsou uvedeny v souboru `ivttests.txt` na platformách Windows a `ivttests_unix.txt` na platformách UNIX and Linux .

Související informace

runivt: IBM MQ přenos pro ověřovací test instalace protokolu SOAP

Vyvíjení webových služeb pro přenos IBM MQ pro protokol SOAP

Použijte normální vývojové prostředí webové služby pro vývoj služeb pro použití s přenosem IBM MQ pro SOAP.

Než začnete

1. Plánujete-li používat nástroje příkazového řádku dodávané s přenosem IBM MQ pro protokol SOAP, postupujte takto:
   a. Vytvořte adresář implementace pro službu.
   b. Spusťte příkazové okno v adresáři.
   c. Pro .NET, csc.exe a wsdl.exe musí být v cestě a musí být ze stejné verze produktu .NET Framework.
   d. Pro produkt Java:
      i) Spuštěním příkazu **amqwsetcp** nastavte cestu ke třídě.
      ii) Prostředí JRE produktu IBM a sada JDK na stejné úrovni verze musí být v aktuální cestě. Úroveň verze musí být nejméně 5.0.
      iii) Upravte cestu ke třídě tak, aby obsahovala umístění dalších knihoven produktu `.jar` a adresářů obsahujících soubory .java, včetně balíků pro vývoj, který vyvíjíte. Umístěte aktuální adresář `"."` do cesty ke třídě.
      iv) Vytvořte adresář vzhledem k aktuálnímu adresáři v okně příkazového řádku, který odpovídá názvu balíku služby, kterou vyvíjíte.
2. Případně můžete použít nástroje pracovní plochy, které podporují vývoj webových služeb. Ukázkové úlohy vývoje používají produkt Microsoft Visual Studio 2008, Eclipse IDE for Java EE Developers and WebSphere Application Server Community Edition.

Informace o této úloze

Existující webové služby nepotřebují žádnou úpravu pro práci s přenosem WebSphere pro protokol SOAP. Nástroje dodávané s přenosem produktu IBM MQ pro implementaci protokolu SOAP implementují webovou službu a spouštějí ji s použitím modulu listener SOAP produktu IBM MQ . The tools also generate WSDL, .NET client stubs, and `.java` proxy classes to develop IBM MQ transport for SOAP clients.

Postupujte takto, chcete-li vytvořit službu a připravit ji na implementaci a generování klientů. Chcete-li vytvořit službu pomocí produktu Eclipse nebo Microsoft Visual Studio 2008, postupujte podle kroků souvisejících s úlohami.

Postup

1. Vyvinout službu pomocí běžného vývojového prostředí.
2. Testování služby pomocí klienta webových služeb HTTP
3. Chcete-li připravit adresář implementace, postupujte takto:
   - NapříkladJava
     a. Zkopírujte soubor `.java` , který definuje rozhraní služby, do adresáře implementace.
     b. Zkopírujte všechny soubory produktu `.class` pro danou službu do adresáře, který odpovídá názvu balíku.
     c. Zkontrolujte, zda cesta ke třídě může vyhledat všechny požadované třídy: zkompilujte soubor služby `.java` pomocí produktu **javac**.
   - Například.NET
     a. Copy the `.asmx` file defining the service into the deployment directory, and
     b. Používáte-li model s kódem, zkopírujte všechny soubory `.dll` do adresáře *deployment directory*`\bin` .

Vývoj služby .NET 1 nebo 2 pro přenos IBM MQ pro SOAP pomocí produktu Microsoft Visual Studio 2012

Vyvíjejte webovou službu SampleStockpro .NET 1 nebo .NET 2 pomocí produktu Microsoft Visual Studio 2012.

Informace o této úloze

Vytvořte službu StockQuote s použitím kódu za použití produktu Microsoft Visual Studio 2012.

Postup

1. Vytvořte šablonu pro službu a zkontrolujte, zda je spuštěna na protokolu HTTP.
   a) Start Visual Studio 2012 > **Soubor** > **Nový** > **Projekt ...**. Vyberte volbu **C#** Typ projektu, **, NET Framework 2**a **Aplikace webové služby ASP.NET**. Zadejte **Název:** a **Název řešení:** `StockQuoteDotNet` > **OK**
   b) Pravým tlačítkem myši klepněte na položku **Service1.asmx** v části **Průzkumník řešení** > **Přejmenovat** > `StockQuote.asmx`.
   c) Změňte fragment kódu `public class Service1` na `public class StockQuote`.
   d) Pravým tlačítkem myši klepněte na položku **StockQuote.asmx** v **Průzkumníku řešení** > **Otevřít pomocí ...** > **Editor XML**. Změnit `Class="StockQuoteDotNet.Service1"` na `Class="StockQuoteDotNet.StockQuote"`
   e) Změňte fragment kódu `[WebService(Namespace = "http://tempuri.org/")]` na `[WebService(Namespace = "http://stock.samples/")]`.
   f) Odeberte řádek kódu `[ToolboxItem(false)]`.
   g) Zkontrolujte vše, co je zatím správné: **Debug** > **Start Debugging (F5)**. Ověřte výstup v Průzkumníku.
2. Přidejte metody z ukázky `SQDNNonInline.asmx.cs`a otestujte službu na HTTP.
   a) Otevřete produkt *MQ_INSTALLATION_PATH* `\tools\soap\samples\dotnet\SQDNNonInline.asmx.cs` a nahraďte metodu `HelloWorld` čtyřmi metodami `Quote` ; viz Obrázek 167 na stránce 1271. *MQ_INSTALLATION_PATH* představuje adresář, kde je nainstalován produkt IBM MQ .
   b) **Sestavit** > **Znovu sestavit** řešení > Klepněte pravým tlačítkem myši na jeden z **Podproces** v chybě > **Vyřešit** > Použití **System.Threading**.
   c) Stisknutím klávesy F5 spustíte ladění programu.

      Služba není souhlasná s profilem WS-I Basic Profile v1.1. Máte volbu buď změnit anotaci WebMethod z `[SoapRpcMethod]` na `[SoapDocumentMethod]`, nebo odebrat anotaci `[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]`.
   d) Stisknutím klávesy F5 ověřte implementaci pomocí protokolu HTTP.
3. Generujte kód WSDL, klienty a spusťte službu pomocí transportu produktu IBM MQ pro protokol SOAP.
   a) Otevřete příkazové okno ve stromu adresáře projektu, kde je uložen `StockQuote.asmx` .
   b) (Volitelné) Chcete-li generovat artefakty, použijte amqwdeployWMQService . Správce front musí být spuštěn:

      ```
      amqswdeployWMQService -f StockQuote.asmx
      -u "jms:/queue?initialContextFactory=com.ibm.mq.jms.Nojndi
      &connectionFactory=()
      &destination=REQUESTDOTNET@QM1
      &targetService=StockQuote.asmx"
      StockQuote.asmx StockQuote.wsdl
      ```

      Všechny artefakty se vytvoří ve stromu adresáře `./generated` .
   c) (Volitelné) Generujte pouze WSDL pro volání služby pomocí přenosu IBM MQ pro SOAP.

      ```
      amqswsdl -u "jms:/queue?initialContextFactory=com.ibm.mq.jms.Nojndi
      &connectionFactory=()
      ```

```
&destination=REQUESTDOTNET@QM1
&targetService=StockQuote.asmx"
StockQuote.asmx StockQuote.wsdl
```

a) Spusťte modul listener produktu .NET . Buď použijte `.\generated\server\startWMQNListener.cmd` , nebo zadejte příkaz:

```
amqSOAPNETListener -u "jms:/queue?initialContextFactory=com.ibm.mq.jms.Nojndi
&connectionFactory=()
&destination=REQUESTDOTNET@QM1
&targetService=StockQuote.asmx"
```

4. Otestujte službu pomocí klienta generovaného ze souboru WSDL nebo pomocí klientů vygenerovaných produktem **amqwdeployWMQService**.

Ukázkový kód

Ukázková webová služba .NET , `StockQuoteDotNet`, je nainstalována v produktu *MQ_INSTALLATION_PATH* `\tools\soap\samples\dotnet`. *MQ_INSTALLATION_PATH* je adresář, kde je nainstalován produkt IBM MQ . Vazba webových služeb publikovaných ukázek se mírně liší od vazby použité v rámci úlohy. Úloha používá předvolby použité v produktu Microsoft Visual Studio 2012.

Existují dva příklady webových služeb .NET Framework 1 a .NET Framework 2. `StockQuoteDotNet.asmx`, je vložená služba. `SQDNNoninline.asmx`, je webová služba implementovaná kódem, implementovaná serverem `SQDNNoninline.asmx.cs`.

`StockQuoteDotNet` má čtyři metody:

1. `float getQuote(String symbol)`
2. `void getQuoteOneWay(String symbol).`
3. `int asyncQuote(int delay)`
4. `float getQuoteDOC(String symbol)`

```
<%@ WebService Language="C#" Class="StockQuoteDotNet" %>
using System;
using System.Web.Services;
using System.Web.Services.Protocols;
using System.Web.Services.Description;
using System.Threading;
[WebService (Namespace="http://stock.samples")]
public class StockQuoteDotNet {
     [WebMethod] [ SoapRpcMethod(OneWay=true) ]
     public void getQuoteOneWay(String symbol) {
          if (symbol.ToUpper().Equals("DELAY")) Thread.Sleep(5000);
          System.Console.WriteLine("getQuoteOneWay was invoked.");
     }
     [WebMethod] [SoapRpcMethod]
     public float getQuote(String symbol) {
          if (symbol.ToUpper().Equals("DELAY")) Thread.Sleep(10000);
          return 88.88F;
     }
     [WebMethod] [SoapRpcMethod]
     public int asyncQuote(int delay) {
          Thread.Sleep(delay);
          return delay;
     }
     [WebMethod]
     public float getQuoteDOC(String symbol) {
          return 77.77F;
     }
}
```

Obrázek 167. Vložená služba: `StockQuoteDotNet.asmx`

```
<%@ WebService Language="C#" Codebehind="SQDNNonInline.asmx.cs" Class="SQDNNonInline" %>
```

Obrázek 168. Kód-za: Návrh `SQDNNonInline.asmx`

```
using System;
using System.Web.Services;
using System.Web.Services.Protocols;
using System.Web.Services.Description;
using System.Threading;

[WebService(Namespace = "http://stock.samples")]
public class SQDNNonInline : System.Web.Services.Protocols.SoapHttpClientProtocol
{

    [WebMethod]
    [SoapRpcMethod(OneWay = true)]
    public void getNonInlineQuoteOneWay(String symbol)
    {
        if (symbol.ToUpper().Equals("DELAY")) Thread.Sleep(5000);
        System.Console.WriteLine("getNonInlineQuoteOneWay was invoked.");
    }

    [WebMethod]
    [SoapRpcMethod]
    public float getNonInlineQuote(String symbol)
    {
        if (symbol.ToUpper().Equals("DELAY")) Thread.Sleep(10000);
        return 88.88F;
    }

    [WebMethod]
    [SoapRpcMethod]
    public int asyncNonInlineQuote(int delay)
    {
        Thread.Sleep(delay);
        return delay;
    }
    [WebMethod]
    public float getNonInlineQuoteDOC(String symbol)
    {
        return 77.77F;
    }
}
```

Obrázek 169. Kód-za: Implementace: `SQDNNonInline.asmx.cs`

Související úlohy

Vývoj webové služby JAX-WS pro W3C SOAP přes JMS

Webová služba, která je svázána s doporučujícím doporučením W3C pro protokol SOAP over JMS , musí být spuštěna v kontejneru EJB aplikačního serveru JEE . Tato úloha je krokem 2 připojení klienta webové služby Axis2 a webové služby implementované na serveru WebSphere Application Server pomocí protokolu SOAP W3C přes JMS .

Vývoj webové služby JAX-WS pro W3C SOAP přes JMS

Webová služba, která je svázána s doporučujícím doporučením W3C pro protokol SOAP over JMS , musí být spuštěna v kontejneru EJB aplikačního serveru JEE . Tato úloha je krokem 2 připojení klienta webové služby Axis2 a webové služby implementované na serveru WebSphere Application Server pomocí protokolu SOAP W3C přes JMS .

Než začnete

Pomocí produktu Rational Application Developer vytvořte webovou službu EJB. Průvodce webovými službami v produktu Rational Application Developer má možnost vytvořit webovou službu pomocí doporučeného doporučení W3C pro vazbu SOAP over JMS . Produkt Rational Application Developer 7.54

je povinný. Cvičení používá produkt Rational Application Developer zahrnutý v produktu Rational Software Architect for WebSphere Software v7.5.5.1,

Objekt EJB je implementován do produktu WebSphere Application Server z produktu Rational Application Developer jako součást této úlohy.

Chcete-li vytvořit WSDL skutečně použité v úloze, musíte nejprve nastavit profil Liberty. Poté můžete buď importovat WSDL z dynamického webového projektu v pracovním prostoru Galileo platformy Eclipse , nebo ze spuštěné webové služby HTTP implementované do profilu Liberty.

WebSphere Application Server může být stále spuštěn. Pokud tomu tak není, můžete jej spustit ze zobrazení Servery v RAD.

Informace o této úloze

In this task, you redeploy the `StockQuoteAxis` service from running as a JAX-RPC Axis service run by the **`SimpleJavaListener`** using IBM MQ transport for SOAP, to being a JAX-WS service running in WebSphere Application server using the W3C SOAP over JMS protocol.

Existují dvě části pro migraci služby z **`SimpleJavaListener`** na WebSphere Application Server:

1. Vygenerujte rozhraní webové služby ze souboru WSDL pro danou službu pomocí průvodce webovou službou Topdown EJB v produktu Rational Application Developer.
2. Implementace služby naimportováním ukázky IBM MQ SOAP `StockQuoteAxis.java`.

Alternativním přístupem by bylo generování služby zdola nahoru, ze serveru `StockQuoteAxis.java`. Chcete-li však mít jistotu, že rozhraní migrované služby je identické, je přístup shora dolů lepší, protože používá stejný WSDL.

Webová služba je vyvinuta pro kontejner EJB a ne pro webový kontejner, protože podpora produktu JMS je součástí kontejneru EJB.

Postup

1. Spusťte produkt Rational Application Developer a ověřte, že je portál WebSphere Application Server spuštěn.
   a) Spusťte produkt Rational Application Developer v novém pracovním prostoru.
   b) Otevřete perspektivu produktu Java EE.
   c) Otevřete kartu **Servery** a zkontrolujte, zda je spuštěn portál WebSphere Application Server .
      - Pokud v pohledu není k dispozici žádná položka WebSphere Application Server 7.0 , klepněte pravým tlačítkem myši v zobrazení > **Nový** > **Server**. Postupujte podle voleb v průvodci a vytvořte instanci produktu WebSphere Application Server 7.0 .
      - Je-li server přítomen, ale není spuštěn, spusťte jej klepnutím na šipku se šipkou.
      - Chcete-li ověřit vlastnosti a získat rychlý přístup k protokolům serveru, klepněte pravým tlačítkem myši na **WebSphere Application Server 7.0 na lokálním hostiteli** > **Vlastnosti** > **WebSphere Application Server**.
      - Chcete-li spravovat server, buď použijte externí prohlížeč a otevřete adresu URL, `http://localhost:9061/ibm/console/unsecureLogon.jsp`, nebo klepněte pravým tlačítkem myši na **WebSphere Application Server 7.0 na lokálním hostiteli** > **Spustit administrativní konzolu**.
      - Výchozí nastavení je publikovat automaticky. Mnoho lidí preferuje ruční nasazení aktualizací na server. Poklepejte na **WebSphere Application Server 7.0 na lokálním hostiteli**a rozbalte prvek **Publikování** v okně **Přehled** . Klepněte na tlačítko **Nikdy nepublikovat automaticky**.
      - Další výchozí hodnotou, kterou byste mohli chtít změnit, je zrušit zaškrtnutí políčka **Ukončit práci serveru při vypnutí pracovní plochy** v okně **Přehled** .
2. Vytvořit projekty produktu JEE

   Musíte vytvořit projekt Enterprise Application Project (EAR) a projekt EJB (Enterprise Java Bean).

   a) **Soubor** > **Nový** > **Projekt podnikové aplikace**. Pojmenujte projekt `W3CJMSEAR` > **Dokončit**.

Výchozí hodnoty musí identifikovat WebSphere Application Server 7.0 jako cílové běhové prostředí a verzi EAR 5.0. Musí být vybrána výchozí konfigurace.

b) **Soubor** > **Nový** > **Projekt EJB**. Pojmenujte projekt `W3CJMSEJB`. Vyberte volbu `W3CEARJMS` jako **Název projektu EAR** > **Další**.

Výchozí verze modulu EJB je 3.0 a znovu se použije výchozí konfigurace.

c) Zrušte zaškrtnutí políčka **Vytvořit modul JAR klienta EJB** > **Dokončit**.

3. Vygenerujte a implementujte webovou službu EJB ze souboru WSDL produktu `StockQuoteAxis` .

a) **Spustit** > **Spustit průzkumník webových služeb**.

b) Vyberte stránku WSDL pomocí ikon v okně **Průzkumník webových služeb** > klepněte na položku **WSDL main** v modulu Navigator.

c) V okně **Akce** zadejte nebo přejděte na adresu URL WSDL pro produkt `StockQuoteAxis.wsdl`.

Máte-li Liberty spuštěný s produktem `StockQuoteAxis` implementovaným jako služba HTTP, adresa URL je:

```
http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis?wsdl
```

Máte-li v systému souborů WSDL, adresa URL může být:

```
File:\Dirpath\StockQuoteAxis\WebContent\wsdl\StockQuoteAxis.wsdl
```

d) Klepněte na řádek obsahující importovanou adresu URL ve stromu Navigator .

Jedná se o řádek bezprostředně následující po **WSDL-hlavní**, pokud se jedná o první WSDL, který jste importovali do průzkumníku webových služeb.

e) V okně **Akce** klepněte na volbu **Spustit průvodce webovou službou** > **Skepleton webové služby** > **Spustit** .

f) V průvodci webovou službou vyberte volbu **Nejvyšší webová služba EJB** .

Vyberte nebo ověřte konfiguraci pomocí informací z volby Tabulka 190 na stránce 1274 Zkontrolovat **přepsat soubory bez varování** > **Další**.

Tabulka 190. Konfigurace webové služby shora dolů

Pole	**Hodnota**
Server	`WebSphere Application Server 7.0`
Doba běhu webové služby	`IBM WebSphere JAX-WS`
projekt služby	`W3CJMSEJB`
Projekt EAR služby	`W3CJMSEAR`
Konfigurace:	`No client generation`

g) Na stránce pod titulkem **Zadejte volby pro vytvoření webové služby WebSphere JAX-WS-EJB shora dolů**zaškrtněte políčko **Přepnout na vazbu JMS** . Zkontrolujte také volbu **Povolit styl obalu**, **Kopírovat WSDL do projektu**a **Generovat deskriptor implementace webové služby** > **Další**.

h) Na stránce s názvem **Konfigurace vazby JMSWebSphere JAX-WS**zaškrtněte políčko **Použít protokol interoperability SOAP/JMS** a zadejte hodnoty z Tabulka 191 na stránce 1274, ponechejte ostatní pole prázdná > **Další**.

Tabulka 191. Konfigurace vázání WebSphere JAX-WS JMS

Pole	**Hodnota**
Cíl JMS	`queue`
Název rozhraní JNDI cíle:	`requestaxis`

Tabulka 191. Konfigurace vázání WebSphere JAX-WS JMS (pokračování)	
Pole	**Hodnota**
Továrna připojení produktu JMS	`qm1`
Odpovědět na jméno	`W3CJMSEAR`
Konfigurace:	`replyaxis`

a) Na stránce s názvem **Konfigurace projektu směrovače WebSphere JAX-WS**zadejte `qm1as` do pole **Název JNDI ActivationSpec** > **Další**.

RAD trvá asi 30 sekund na minutu pro generování a nasazení projektu.

b) Ignorujte volby na stránce **Publikace webové služby** > **Dokončit**.

4. Zkontrolujte vygenerovaný kód WSDL.

Požádali jste o vygenerování a uložení kódu WSDL specifického pro službu v projektu.

a) V navigátoru podnikového průzkumníka otevřete složku **W3CJMSEJB** > **ejbmodule** > **META-INF** > **wsdl**. Poklepáním na ikonu `StockQuoteAxis.wsdl` ji otevřete v editoru WSDL.

Zkontrolujte vazbu. Zobrazí se adresa URL produktu JMS :

```
jms:jndi:requestaxis?jndiConnectionFactoryName=qm1&targetService=StockQuoteAxis
```

5. Volitelný krok: Svázání EJB s protokolem SOAP prostřednictvím protokolu HTTP pomocí rozhraní JAX-WS.

Poskytnutí dvou vazeb k sadě EJB poskytuje klientům volbu vazeb SOAP pro volání webové služby. Poskytuje klientům také prostředky k dotazování na webový server za účelem získání jeho kódu WSDL pomocí protokolu HTTP.

Kroky pro vytvoření vazby EJB na SOAP přes protokol HTTP nejsou zahrnuty jako součást úlohy.

6. Implementovat a znovu implementovat produkt `StockQuoteAxis` s použitím ukázky `StockQuoteAxis.java`

a) V navigátoru podnikového průzkumníka otevřete složku **W3CJMSEJB** > **Services** Poklepejte na `StockQuoteAxisService` , abyste otevřeli implementační třídu v editoru Java .

b) Otevřete ukázkový program `StockQuoteAxis.java` ve složce *`WebSphere MQ Installation directory`*`\tools\soap\samples\java\server` > Vyberte všechny metody, ale ne název třídy > **Kopírovat**.

c) V produktu `StockQuoteAxisSoapBindingImpl.java` vyberte všechny metody, nikoli však název třídy, a vložte je do metod z produktu `StockQuoteAxis.java`.

d) Přidejte tiskový příkaz na výstup do konzoly WebSphere Application Server , když se zavolá služba. Změňte metodu getQuote(symbol řetězce):

```
public float getQuote(String symbol) {
System.out.println("StockQuoteAxisSoapBindingImpl called with symbol: "
+ symbol);
return ((float) 55.25);
```

e) Opravte importy: **Zdroj** > **Uspořádat importy** > **Uložit**.

f) Opravte tyto tři chyby v důsledku implementace, která neodpovídá rozhraní.

Chyby jsou způsobeny třemi z metod v části `StockQuoteAxis.java` výjimek a kódem WSDL pro službu, která neobsahuje žádné zprávy o poruchách. Problém je diagnostikován jako nesoulad mezi podpisy metody a anotacemi webové služby metody.

Buď anotujte metody pomocí znaku @WebFault a znovu vygenerujte WSDL, nebo zachovejte rozhraní beze změny a odeberte výjimky.

Chcete-li zachovat rozhraní stejné, odeberte tři `throws exception` z podpisů metod > **Uložit**.

Jak pokračovat dále

“Implementace na klienta Axis2 pomocí protokolu W3C SOAP prostřednictvím produktu JMS” na stránce 1316

Související úlohy

Vývoj služby .NET 1 nebo 2 pro přenos IBM MQ pro SOAP pomocí produktu Microsoft Visual Studio 2012
Vyvíjejte webovou službu SampleStockpro .NET 1 nebo .NET 2 pomocí produktu Microsoft Visual Studio 2012.

Vývoj klientů webových služeb IBM MQ pro produkt IBM MQ pro protokol SOAP

Použijte své normální vývojové prostředí pro vývoj klientů webových služeb pro použití s přenosem IBM MQ pro SOAP.

Než začnete

Vytvořte službu. Můžete použít jeden z příkladů v produktu “Vyvíjení webových služeb pro přenos IBM MQ pro protokol SOAP” na stránce 1269.

Vyberte volby, jak vyvíjet, implementovat a používat klienty, a kde získat WSDL pro generování klienta.

Rozhodněte se pro váš přístup k vývoji klientů a služeb pro přenos IBM MQ pro protokol SOAP.

Existují dva přístupy.

1. Použijte standardní vývojové nástroje, vyvinout službu HTTP a klienta a potom použít adresu URL pro přenos WebSphere MQ pro SOAP.
2. Použijte nástroje a ukázky dodané s přenosem IBM MQ pro protokol SOAP.

Pokud použijete trasu HTTP, můžete spustit službu na serveru HTTP a také ji spustit pomocí přenosu IBM MQ pro protokol SOAP. Chcete-li ji spustit pomocí transportu produktu IBM MQ pro protokol SOAP, nakonfigurujte příslušný modul listener produktu IBM MQ pro protokol SOAP a nastavte cesty a deskriptory implementace pro spuštění služby. Konfigurace pro SOAP poskytuje nástroje poskytované produktem IBM MQ pro konfiguraci SOAP. Volitelně můžete prostředí nakonfigurovat tak, aby spouštěly listenery.

Nástroje dodávané s přenosem produktu IBM MQ pro SOAP jsou užitečné při zahájení a učení se způsobu implementace přenosu. Pro provozní práci je výhodné používat standardní nástroje a implementovat stejnou službu přístupnou pro různé transporty SOAP.

Rozhodněte se, jaký typ klienta se má vyvíjet

Je třeba rozhodnout, jaký typ klienta webové služby se má vyvíjet. Volba závisí na tom, zda znáte rozhraní služby a adresu služby.

Je-li rozhraní známo, použijte nástroje Axis nebo .NET ke generování tříd klienta proxy z rozhraní služby. Třídy klientů proxy zjednodušují zápis klienta pro volání služby. Je-li umístění služby známé při vývoji klienta, použijte statické rozhraní proxy. Pokud se změní umístění služby, například pokud je služba znovu implementována na produkční server, použijte dynamické rozhraní serveru proxy.

Není-li rozhraní služby známo v době, kdy vyvíjíte klienta, na ose Axis, můžete vytvořit klienta DII (Dynamic Invocation Interface) pro Axis 1.4. Klient DII používá generické rozhraní k volání libovolné služby. Chcete-li předat parametry konkrétní službě správně, musíte sestavit specifické rozhraní služby programově. Sestavte rozhraní programově v klientovi nebo načítáním WSDL pro danou službu do klienta. V prostředí Axis2můžete vytvořit klienta Dispatch. Klient Dispatch používá model dokumentu k popisu požadavku klienta, zatímco klient DII používá model volání. Obě pracují na sestavení požadavku dynamicky.

Získat WSDL pro službu

S výjimkou případu, kdy se rozhraní služby sestavuje programově, musíte nejprve získat WSDL služby, abyste vytvořili klienta webové služby. Služba WSDL je možné získat ze tří různých zdrojů:

1. Přímo z implementace webové služby pomocí nástroje, jako je **java2wsdl** (Axis) nebo **disco** (.NET).
2. Dotazem na webovou službu pomocí adresy URL: *Web service http url* ?wsdl.
3. Ze souboru, buď na systému souborů, nebo z registru, jako je registr UDDI nebo produkt WebSphere Service Registry and Repository.

Poznámka: Není-li služba přístupná pomocí protokolu HTTP, pak dotaz WSDL nefunguje. Samotná služba může být k dispozici pouze prostřednictvím transportu produktu IBM MQ pro protokol SOAP.

Kód WSDL generovaný produktem **amqwdeployWMQService** není stejný jako soubor WSDL generovaný pomocí produktu **java2wsdl** nebo **disco**. Vygenerovaný kód WSDL se také liší od všech WSDL, které jste mohli začít s vytvořením služby "Top Down". Na ose Axis mapuje deskriptor implementace produktu `server-config.wsdd` zprávu SOAP vytvořenou klientem na operaci a službu. Produkt **amqwdeployWMQService** generuje jiný deskriptor implementace z platformy Eclipse.

WSDL, který používáte k sestavení klientů, závisí na způsobu implementace služby:

Implementováno pomocí amqwdeployWMQService
Použijte kód WSDL generovaný produktem **amqwdeployWMQService**. Zadejte příznak -w a vyberte volbu `rpcLiteral` WSDL. Z důvodů kompatibility můžete vybrat volbu `rpcEncoded` WSDL. `rpcEncoded` WSDL funguje pouze s klienty .NET a Axis 1.4 .

Ruční implementace pomocí produktu SimpleJavaListener
Použijte jeden z následujících souborů WSDL:

1. Kód WSDL použitý k definování služby nebo uložení v úložišti.
2. Jazyk WSDL generovaný ze služby produktem **java2wsdl**.
3. Dotazovaný kód WSDL s použitím adresy URL *Web service http url* ?wsdl, pokud je k dispozici na serveru HTTP. Chcete-li importovat definici služby přímo do platformy Eclipse, můžete spustit nástroj, jako např. průzkumník webových služeb.

Možná budete muset změnit identifikátor URI pro službu. Změňte ji z adresy služby HTTP na identifikátor URI pro přenos IBM MQ pro SOAP.

Ruční implementace pomocí produktu amqSOAPNETListener.
Použijte jeden z následujících souborů WSDL:

1. Kód WSDL použitý k definování služby nebo uložení v úložišti.
2. Kód WSDL získaný ze třídy služeb .NET (.asmx). pomocí produktu **disco**.
3. Dotazovaný kód WSDL s použitím adresy URL *web services http url* ?wsdl, pokud je k dispozici. Chcete-li importovat definici služby přímo do platformy Eclipse, můžete spustit nástroj, jako např. průzkumník webových služeb.
4. Kód WSDL získaný spuštěním produktu **amqswsdl** proti třídě služeb produktu .NET (.asmx).

Možná budete muset změnit identifikátor URI pro službu. Změňte ji z adresy služby HTTP na identifikátor URI pro přenos IBM MQ pro SOAP.

Nasazeno na Windows Communication Foundation
Službu WSDL získejte pomocí adresy URL *Web service http url* ?wsdl. Služba musí být definována pomocí konfigurace chování `serviceMetaData` jako součást definice služby.

Implementace na jinou platformu serveru.
Postupujte podle pokynů poskytnutých spolu s platformou o tom, jak získat správný kód WSDL služby.

Informace o této úloze

Vyvíjejte klienty pomocí standardních vývojových nástrojů. Následující úlohy znázorňují, jak sestavit klienty pro produkty .NET 1 a 2, Axis 1.4 (JAX-RPC) a Axis2 (JAX-WS). Pro produkt Windows Communication Foundation si prohlédněte odkazy na související úlohy.

Vývoj klienta JAX-RPC pro produkt WebSphere transport pro protokol SOAP pomocí prostředí Eclipse

Vyviňte klienta webové služby Axis 1.4 , který má být spuštěn pomocí přenosu produktu IBM MQ pro protokol SOAP.

Než začnete

Musíte mít k dispozici službu. Musíte mít spuštěný aplikační server na platformě Eclipse, která podporuje webové služby Axis 1.4 . V této úloze používáme volně dostupný profil Liberty. Můžete také použít Tomcat 6, což je menší aplikační server s otevřeným zdrojovým kódem.

Informace o této úloze

Tato úloha ukazuje vývoj tří typů klientů pro ukázkovou službu Axis StockQuotes použitím platformy Eclipse spuštěnou na serveru Windows. Klienti jsou statický a dynamický klient vyvinutý pomocí klienta proxy klienta a klienta DII.

Jsou ilustrovány dva alternativní přístupy ke generování proxy klienta z WSDL:

1. Generování proxy klienta pomocí produktu **amqwdeployWMQService**.
2. Import WSDL do platformy Eclipsea použití průvodce webovou službou pro generování proxy klienta.

Postup

1. Spusťte vývojáři Eclipse IDE pro vývojáře Java EE.
2. Vytvořte projekt Java s názvem `StockQuoteAxisClient`:
   a) Přepněte do perspektivy produktu Java > **Soubor** > **Nový** > **Projekt produktuJava**. V poli **Project name** na typu **Vytvořit stránku projektu Java** `StockQuoteAxisEclipseClient`. Ujistěte se, že prováděcí prostředí je buď **J2SE1-1.4** , nebo **J2SE-1.5** > **Další**.
   b) Na stránce **Nastavení prostředí Java** vyberte kartu **Knihovny** > **Přidat externí soubory JAR ...**
   c) Přejděte na *MQ_INSTALLATION_PATH*`/java/lib` a vyberte všechny soubory `.jar` > **Otevřít**.
   MQ_INSTALLATION_PATH je adresář, kde je nainstalován produkt IBM MQ .
   d) Přejděte na *MQ_INSTALLATION_PATH*`/java/lib/soap` a vyberte všechny soubory `.jar` > **Otevřít**. Musíte mít nainstalovaný produkt `axis.jar` z instalačního média produktu IBM MQ do tohoto adresáře.
   MQ_INSTALLATION_PATH je adresář, kde je nainstalován produkt IBM MQ .
   e) Karta **Knihovna** nyní odkazuje na všechny soubory `.jar` potřebné k sestavení klienta > **Dokončit**.
3. Postupujte podle jednoho z těchto dvou přístupů k vytvoření serverů proxy v prostředí Eclipse pro ukázkovou webovou službu Axis StockQuote:
   - Generujte proxy klienta pomocí produktu **amqwdeployWMQService**.
     a. Vytvořte správce front. Pro úlohu create `QM1` jako výchozího správce front.
     b. Vytvořte pracovní adresář, `samples`. Zkopírujte ukázkový program `StockQuoteAxis.java` do produktu `samples/soap/server`.
     c. Upravte `amqwsetcp.cmd` v produktu *MQ_INSTALLATION_PATH*`/bin` tak, aby obsahoval aktuální adresář v cestě ke třídě. *MQ_INSTALLATION_PATH* je adresář, kde je nainstalován produkt IBM MQ .
     d. Otevřete příkazové okno v produktu `samples` a spusťte upravený příkaz **amqwsetcp** .
     e. Spuštěním příkazu vytvořte WSDL pro službu Axis StockQuote.

```
amqwdeployWMQService -f soap/server/StockQuoteAxis.java -c genAxisWsdl
-u "jms:/queue?destination=REQUESTAXIS
&initialContextFactory=com.ibm.mq.jms.Nojndi
&connectionFactory=(connectQueueManager(QM1)binding(auto))"
```

Zapamatujte si: Použijte "`/`", spíše než "`.`" nebo "`\`" , když používáte příkazy Java .

Tip: Místo importu vygenerovaných serverů proxy do platformy Eclipsemůžete importovat generovaný WSDL z produktu `.samples/generated`. Výsledné servery proxy se liší dvěma způsoby:

i) Názvy balíků se liší-které můžete refaktorovat.

ii) Servery proxy generované produktem Eclipse obsahují přídavnou třídu pomocníka `StockQuoteAxisProxy.java`

f. Vytvořte proxy klienta pro službu Axis StockQuotetak, že spustíte příkaz:

```
amqwdeployWMQService -f soap/server/StockQuoteAxis.java -c genProxiestoAxis
-u "jms:/queue?destination=REQUESTAXIS
&initialContextFactory=com.ibm.mq.jms.Nojndi
&connectionFactory=(connectQueueManager(QM1)binding(auto))"
```

g. Naimportujte proxy klienta do StockQuoteAxisClient:

i) Pravým tlačítkem myši klepněte na položku **StockQuoteAxisClient\src** > Vyberte **Systém souborů** > **Další** > **Procházet ...** > najděte složku `.\samples\generated\client\remote\soap\server` > **OK**.

ii) Zkontrolujte **server** na stránce **Import** > **Dokončit**.

h. Refaktorujte název balíku na `soap.server`.

i) Klepněte pravým tlačítkem myši na balík obsahující proxy klienta > **Refaktorovat** > **Přejmenovat**. Zadejte **New name**: `soap.server` > ponechejte vybrané předvolby pro ostatní volby > **OK**. Všechny chyby jsou opraveny.

- Generujte proxy klienta pomocí Eclipse.

Máte volbu způsobů, jak získat kód WSDL pro službu. V tomto příkladu byla služba implementována do profilu Liberty Profile a vy získáte soubor WSDL z webového serveru.

a. V prostředí Eclipsepřepněte na webovou perspektivu a zkontrolujte, zda je spuštěn profil Liberty a že je implementována a synchronizována osa Axis StockQuote.

b. Naimportujte WSDL do průzkumníku webových služeb:

i) Klepněte na ikonu **Průzkumník webových služeb** na řádku s akcemi nebo klepněte na volbu **Spustit** > **Spustit průzkumník webových služeb**.

ii) Klepněte na ikonu stránky WSDL v průzkumníku webových služeb a přepněte se na stránku WSDL.

iii) Klepněte na volbu **Hlavní soubor WSDL** v okně Navigator v průzkumníku webových služeb.

iv) Zadejte adresu URL webové služby a za ní klepněte na tlačítko ?WSDL. Adresa URL pro osu StockQuoteimplementovaná v profilu Libertyje následující:

```
http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis?wsdl
```

c. Generovat proxy klienta:

i) V navigátoru průzkumníku webových služeb klepněte na volbu **http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis?wsdl**.

ii) V okně **Akce** klepněte na volbu **Spustit průvodce webovou službou** >, chcete-li vybrat volbu **Klient webové služby** > **Přejít**.

iii) Na první stránce průvodce klepněte na odkaz projektu **Klient** v konfiguraci > Vyberte projekt klienta **StockQuoteAxisClient** na první stránce > **OK**.

Tip: Okno průvodce může ztratit fokus. Musíš se vrátit do soustředění ručně.

iv) Běhové prostředí webové služby musí být Apache Axis pro generování klienta JAX-RPC.

v) Klepněte na tlačítko **Dokončit**.

vi) Změňte statickou adresu URL služby tak, aby ukazovala na přenos IBM MQ pro adresu SOAP pro službu Axis StockQuote. Můžete zvolit přeskočení tohoto kroku, dokud jste neotestovali klienta se serverem HTTP.

a) Otevřete `StockQuoteAxisServiceLocator.java` a vyhledejte deklaraci pro `StockQuoteAxis_address`.

b) Změnit adresu URL na

```
"jms:/queue?destination=REQUESTAXIS
&amp;initialContextFactory=com.ibm.mq.jms.Nojndi
&amp;connectionFactory=(connectQueueManager(QM1)binding(auto))"
```

Tip: Platforma Eclipse automaticky transformuje & na `&` a naopak, když kopírujete a vložíte řetězce do kódu `.java` .

d. Vytvořte tři třídy klienta produktu Java , přičemž každá z nich má hlavní metodu:

i) Vytvořit balík. Pravým tlačítkem myši klepněte na položku **StockQuoteAxisClient/src** > **Nový balík**. Pojmenujte jej `soap.client` > **Dokončit**.

ii) Vyberte volbu **soap.client** > **New** > **Class**. Pojmenujte třídu `SQAStaticClient` > Zkontrolujte **public static void main (string [] args)** > Dokončit

iii) Zopakujte proceduru pro vytvoření `SQADynamicClient.java` a `SQADIIClient.java`

e. Zapište kód klienta.

Obrázek 173 na stránce 1283 prostřednictvím Obrázek 177 na stránce 1285 poskytují příklady tří stylů kódu klienta. Příklady používají adresu URL HTTP k testování klienta pomocí služby Axis StockQuoteimplementované na server HTTP. Chcete-li spustit klienty na základě služby Axis StockQuoteimplementované pomocí přenosu IBM MQ pro SOAP, změňte adresu URL na:

```
"jms:/queue?destination=REQUESTAXIS
connectionFactory=(connectQueueManager(QM1)binding(auto))
initialContextFactory=com.ibm.mq.jms.Nojndi
targetService=soap.server.StockQuoteAxis.java
replyDestination=SYSTEM.SOAP.RESPONSE.QUEUE"
```

– Produkt Obrázek 173 na stránce 1283 a portál Obrázek 175 na stránce 1284 používají server proxy generovaný platformou Eclipse, která má navíc pomocnou třídu StockQuoteAxisproxy, která usnadňuje kódování kódu.

– Obrázek 174 na stránce 1284 a Obrázek 176 na stránce 1284 používají server proxy generovaný produktem **amqwdeployWMQService**.

– Produkt Obrázek 177 na stránce 1285 nepoužívá žádné třídy serveru proxy.

Každý z klientů volá `com.ibm.mq.soap.Register.extension()` , aby se propojí s přenosem IBM MQ pro protokol SOAP. Rozšíření je registrováno v deskriptoru implementace klienta. Implementace klienta do Axis 1.4 je popsána v tématu “Implementace klienta webové služby do Axis 1.4 pro použití přenosu IBM MQ pro SOAP” na stránce 1311.

f. Spusťte klienty odesláním požadavku SOAP na osu StockQuotehostované serverem WebSphere Application Server Community Edition konfigurovaným v pracovním prostoru.

i) Zkontrolujte, zda je server spuštěn, je implementována a synchronizována položka StockQuoteAxis.

ii) Vyberte nebo otevřete klienta, kterého chcete testovat > Klepněte na tlačítko **Spustit** na řádku s akcemi. Případně klepněte na zelenou ikonu Spustit nebo na osm klepnutí na klienta v navigátoru > **Spustit jako** > **Spustit konfigurace ...**. Konfigurujte parametry, které potřebujete ke spuštění klienta.

g. Spusťte klienta pomocí transportu produktu IBM MQ pro protokol SOAP.

Procedura používá produkt **amqwdeployWMQService** k implementaci služby a pracuje pouze s klientem, který používá WSDL nebo servery proxy sestavené produktem **amqwdeployWMQService**. Chcete-li klienta spustit s použitím původních souborů WSDL nebo serverů proxy sestavených platformou Eclipse, implementujte tuto službu s deskriptorem implementace sestavenou platformou Eclipse. Ruční spuštění produktu **SimpleJavaListener** pomocí názvu vazby portu služby jako názvu `targetServiceName`.

i) Postupujte podle pokynů v části “Implementace služby na Axis 1.4 k použití pro přenos WebSphere pro SOAP pomocí produktu amqwdeployWMQService” na stránce 1304 a implementujte službu do modulu listener protokolu SOAP produktu IBM MQ Simple Java . Implementace služby pracuje pouze pro klienta s použitím serverů proxy WSDL nebo klientů vytvořených produktem **amqwdeployWMQService**.

ii) V příkazovém okně spusťte příkaz **amqwclientconfig** , abyste vytvořili soubor deskriptoru implementace klienta `client-deploy.wsdd`.

iii) Importujte produkt `client-deploy.wsdd` do kořenového adresáře projektu produktu Java , který chcete testovat pomocí transportu produktu IBM MQ pro protokol SOAP.

a) Klepněte pravým tlačítkem myši na projekt Java **StockQuoteAxisEclipseClient** > **Import** > **Systém souborů** > **Další** > **Procházet ...**

b) Přejděte do adresáře, který obsahuje `client-deploy.wsdd` > **Otevřít** > Vyberte adresář na stránce průvodce **Importovat** >, zkontrolujte `client-deploy.wsdd`.

c) Ověřte volbu **Do složky:** zadána hodnota `StockQuoteAxisEclipseClient` > **Dokončit**.

iv) Potvrďte, že pracovní adresář pro spuštění aplikace Java v tomto projektu je adresář `StockQuoteAxisEclipseClient` :

Klepněte pravým tlačítkem myši na projekt produktu Java **StockQuoteAxisEclipseClient** > **Spustit jako ...** > **Spustit konfigurace ...** > Vyberte kartu **(x) = Argumenty** > Ověřte, zda je v pracovním adresáři zaškrtnuto políčko **Výchozí** , a cesta je `StockQuoteAxisEclipseClient`. Případně proveďte jednu z následujících voleb pro výběr jiného umístění nebo souboru obsahujícího konfiguraci klienta:

- Zaškrtněte volbu **Další:** > zadejte cestu k adresáři podle vaší volby.
- V okně **Argumenty VM** zadejte `-Daxis.ClientConfigFile=` *`full path to client deployment descriptor file`*

v) Ujistěte se, že adresa URL je konfigurována tak, aby ukazovala na službu implementovanou pomocí přenosu IBM MQ pro protokol SOAP. Spusťte klienta podle popisu v kroku ii.

Tip: Obvykle se můžete setkat s jednou z těchto chyb:

i) `Exception: No client transport named 'jms' found!` .

ii) Chyba připojení JMS .

iii) `Exception: The AXIS engine could not find a target service to invoke! targetService is soap.server.StockQuoteAxis.java`

iv) `Exception: java.lang.InstantiationException: soap.server.StockQuoteAxis`

Vysvětlení:

i) `client-config.wsdd` nebyl nalezen, nebo zahrnuje řádek `<transport name="jms" pivot="java:com.ibm.mq.soap.transport.jms.WMQSender"/>` v `client-config.wsdd`.

ii) Je možné, že se problém s cestou sestavení-nezahrnuje soubory .jar v produktu *`MQ_INSTALLATION_PATH`*`/java/lib`. *`MQ_INSTALLATION_PATH`* je adresář, kde je nainstalován produkt IBM MQ .

iii) Problém implementace služby, buď s `server-config-wsdd`, nebo s parametry předanými do **SimpleSoapListener**.

iv) Neshoda mezi deskriptorem implementace a implementací služby.

Máte-li potíže se spuštěním klienta v prostředí Eclipse, zkuste použít příkazové okno:

i) Přepněte do adresáře `StockQuoteAxisEclipseClient\bin` ve stromu adresáře pracovního prostoru.

ii) Spusťte **amqwsetcp** a **amqwclientconfig**

iii) Spusťte příkaz `java soap/client/SQAStaticClient`.

Ukázkové klienty webových služeb JAX-RPC

Ukázkové klienty webové služby Java dodané s produktem IBM MQ jsou nainstalovány v produktu *MQ_INSTALLATION_PATH* \tools\soap\samples\java\clients. *MQ_INSTALLATION_PATH* je adresář, kde je nainstalován produkt IBM MQ .

SQAxis2Axis.java
SQAxis2Axis.java, Obrázek 170 na stránce 1282, je dynamický klient proxy pro vyvolání služby StockQuoteAxis . Adresu URL služby, která se má kompilovat do dynamického serveru proxy, můžete přepsat zadáním adresy URL na příkazovém řádku.

SQAxis2DotNet.java
SQAxis2DotNet.java, Obrázek 171 na stránce 1282, je dynamický klient proxy pro vyvolání služby StockQuoteDotNet . Adresu URL služby, která se má kompilovat do dynamického serveru proxy, můžete přepsat zadáním adresy URL na příkazovém řádku.

WsdlClient.java
WsdlClient.java, Obrázek 172 na stránce 1283, je klientem dynamického vyvolání pro vyvolání služby StockQuoteDotNet nebo StockQuoteAxis . Klient vyvolá ve výchozím nastavení službu StockQuoteAxis . Přidejte volbu příkazového řádku -D , vyvolejte službu StockQuoteDotNet a -w pro poskytnutí jiného portu v produktu .\generated\StockQuoteDotNet_Wmq.wsdl .

```
package soap.clients;
import java.net.URL;
import soap.server.*;
public class SQAxis2Axis {
   public static void main(String[] args) {
      com.ibm.mq.soap.Register.extension();
      try {
         StockQuoteAxisService locator = new StockQuoteAxisServiceLocator();
         StockQuoteAxis service = null;
         if (args.length == 0)
            service = locator.getSoapServerStockQuoteAxis_Wmq();
         else
            service = locator.getSoapServerStockQuoteAxis_Wmq(
                new java.net.URL(args[0]));
         System.out.println("Response: " + service.getQuote("XXX"));
      } catch (Exception e) {
         System.out.println("\n>>> EXCEPTION WHILE RUNNING ProxyClient DEMO <<<\n");
                            e.printStackTrace();
         System.exit(2);
      }
  }
```

Obrázek 170. SQAxis2Axis.java

```
public class SQAxis2DotNet {
public static void main(String[] args) {
   com.ibm.mq.soap.Register.extension();
      try {
         StockQuoteDotNet locator = new StockQuoteDotNetLocator();
         StockQuoteDotNetSoap_PortType service = null;
         if (args.length == 0)
            service = locator.getStockQuoteDotNetSoap();
         else
            service = locator.getStockQuoteDotNetSoap(new java.net.URL(
                  args[0]));
         System.out.println("Response: " + service.getQuoteDOC("XXX"));
      } catch (Exception e) {
         System.out.println("\n>>> EXCEPTION WHILE RUNNING ProxyClient DEMO <<<\n");
                            e.printStackTrace();
         System.exit(2);
      }
   }
}
```

Obrázek 171. SQAxis2DotNet.java

```
package soap.clients;
import com.ibm.mq.soap.*;
import org.apache.axis.utils.Options;
import java.net.URL;
import javax.xml.rpc.Call;
import javax.xml.rpc.Service;
import javax.xml.rpc.ServiceFactory;
import javax.xml.namespace.QName;
public class WsdlClient {
public static void main(String[] args) {
      String wsdlService, wsdlPort, namespace, wsdlSource, wsdlTargetURI, s;
      try {
         Register.extension();
         Options opts = new Options(args);
         if (opts.isFlagSet('D') != 0) {
            wsdlService = "StockQuoteDotNet";
            wsdlPort = "StockQuoteDotNetSoap";
            namespace = "http://stock.samples";
            wsdlSource = "file:generated/StockQuoteDotNet_Wmq.wsdl";
         } else {
            wsdlService = "StockQuoteAxisService";
            wsdlPort = "soap.server.StockQuoteAxis_Wmq";
            namespace = "soap.server.StockQuoteAxis_Wmq";
            wsdlSource = "file:generated/soap.server.StockQuoteAxis_Wmq.wsdl";
         }
         if (null != (s = (opts.isValueSet('w'))))
            wsdlPort = s;
         System.out.println("start WsdlClient demo, wsdl port " + wsdlPort
               + " resolving uri to ...");
         QName servQN = new QName(namespace, wsdlService);
         QName portQN = new QName(namespace, wsdlPort);
         Service service = ServiceFactory.newInstance().createService(
               new URL(wsdlSource), servQN);
         Call call = (Call) service.createCall(portQN, "getQuote");
         wsdlTargetURI = call.getTargetEndpointAddress().toString();
         System.out.println("  '" + wsdlTargetURI + "'");
         Object ret = call.invoke(new Object[] { "XXX" });
         System.out.println("Response: " + ret);
      } catch (Exception e) {
         System.out.println("\n>>> EXCEPTION WHILE RUNNING WsdlClient DEMO <<<\n");
         e.printStackTrace();
         System.exit(2);
      }
   }
}
```

Obrázek 172. WsdlClient.java

Ukázkové klienty použité v této úloze:

```
package soap.client;
import soap.server.StockQuoteAxisProxy;
public class SQAStaticClient {
  public static void main(String[] args) {
    try {
      com.ibm.mq.soap.Register.extension();
      StockQuoteAxisProxy sqa = new StockQuoteAxisProxy();
      System.out.println("Static client synchronous result is:"
          + sqa.getQuote("ibm"));
    } catch (Exception e) {
      System.out.println("Exception: " + e);
    }
  }
}
```

Obrázek 173. Statický klient používající server proxy Eclipse

```
package soap.client;
import soap.server.StockQuoteAxis;
import soap.server.StockQuoteAxisService;
import soap.server.StockQuoteAxisServiceLocator;
public class SQAStaticClient {
  public static void main(String[] args) {
    try {
      com.ibm.mq.soap.Register.extension();
      StockQuoteAxisService locator = new StockQuoteAxisServiceLocator();
      StockQuoteAxis sqa = locator.getSoapServerStockQuoteAxis_Wmq();
      System.out.println("Static client synchronous result is: "
          + sqa.getQuote("ibm"));
    } catch (Exception e) {
      System.out.println("Exception: " + e);
    }
  }
}
```

Obrázek 174. Statický klient používající server proxy amqwdeployWMQService

```
package soap.client;
import soap.server.StockQuoteAxisProxy;
public class SQADynamicClient {
  public static void main(String[] args) {
    try {
      com.ibm.mq.soap.Register.extension();
      StockQuoteAxisProxy sqa = new StockQuoteAxisProxy(
          "http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis");
      System.out.println("Dynamic client synchronous result is: "
          + sqa.getQuote("ibm"));
    } catch (Exception e) {
      System.out.println("Exception: " + e);
    }
  }
}
```

Obrázek 175. Dynamický klient používající server proxy generovaný platformou Eclipse

```
package soap.client;

import java.net.URL;
import soap.server.StockQuoteAxis;
import soap.server.StockQuoteAxisService;
import soap.server.StockQuoteAxisServiceLocator;
public class SQADynamicClient {
  public static void main(String[] args) {
    try {
      com.ibm.mq.soap.Register.extension();
      URL sqaURL = new URL(
          "http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis");
      StockQuoteAxisService locator = new StockQuoteAxisServiceLocator();
      StockQuoteAxis sqa = locator.getSoapServerStockQuoteAxis_Wmq(sqaURL);
      System.out.println("Dynamic client synchronous result is: "
          + sqa.getQuote("ibm"));
    } catch (Exception e) {
      System.out.println("Exception: " + e);
    }
  }
}
```

Obrázek 176. Dynamický klient používající server proxy amqwdeployWMQService

```
package soap.client;
import java.net.URL;
import javax.xml.namespace.QName;
import javax.xml.rpc.Call;
import javax.xml.rpc.Service;
import javax.xml.rpc.ServiceFactory;
public class SQADIIClient {
  public static void main(String[] args) {
    try {
      com.ibm.mq.soap.Register.extension();
      URL wsdl = new URL(
          "http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis?wsdl");
      Service SQAService = (ServiceFactory.newInstance()).createService(wsdl,
          new QName("http://server.soap", "StockQuoteAxisService"));
      Call SQACall = SQAService.createCall(new QName("http://server.soap",
          "StockQuoteAxis"), "getQuote");
      System.out.println("DII client synchronous result is "
          + SQACall.invoke(new Object[] { "ibm" }));
    } catch (Exception e) {
      System.out.println("Exception: " + e);
    }
  }
}
```

Obrázek 177. DII klient (bez proxy)

Související úlohy

Vyvíjení klienta JAX-WS pro produkt WebSphere transport pro protokol SOAP s použitím prostředí Eclipse
Vytvořte klienta webové služby Axis2 , který má být spuštěn pomocí přenosu produktu IBM MQ pro protokol SOAP. Ukázkový klient Axis2 dodávaný s přenosem IBM MQ pro protokol SOAP je uveden v seznamu a příkaz **wsimport** použitý ke generování serverů proxy.

Vyvíjení klienta .NET 1 nebo 2 pro přenos WebSphere pro protokol SOAP pomocí produktu Microsoft Visual Studio 2012
Vytvořte klienta webové služby .NET 1 nebo 2, který má být spuštěn pomocí přenosu produktu IBM MQ pro protokol SOAP.

Vyvíjení klienta JAX-WS pro produkt WebSphere transport pro protokol SOAP s použitím prostředí Eclipse

Vytvořte klienta webové služby Axis2 , který má být spuštěn pomocí přenosu produktu IBM MQ pro protokol SOAP. Ukázkový klient Axis2 dodávaný s přenosem IBM MQ pro protokol SOAP je uveden v seznamu a příkaz **wsimport** použitý ke generování serverů proxy.

Než začnete

Získejte knihovny Axis2 a nakonfigurujte vývojový a testovací prostředí ke spuštění klienta.

Poznámka: Pojmenování verzí a vydání používaných Axis způsobuje zmatek. Obvykle Axis 1.4 znamená implementaci JAX-RPC, a Axis2 implementaci JAX-WS.

Axis 1.4 je úroveň verze. Když budete hledat Axis 1.4 na internetu, dostanete se na stránku http://ws.apache.org/axis/. Tato stránka obsahuje seznam předchozích verzí Axix (1.2, 1.3) a finální verzi Axis 1.4 z 22. dubna 2006. Existují i starší vydání Axis 1.4, které opravují chyby, ale ty všechny jsou známy jako Axis 1.4. Jedno z těchto vydání s opravou chyby se dodává s produktem IBM MQ. Pro Axis 1.4 použijte verzi `axis.jar`, která se dodává s produktem IBM MQ, a ne tu, kterou můžete získat ze stránky http://ws.apache.org/axis/.

Webový server Axis se také odkazuje na Axis 1.1, jenž by se měl týkat všech verzí toho, co je častěji nazýváno Axis 1.4. Axis 1.2 se používá pro vše, co se obvykle nazývá Axis2.

Axis 1.5 není novějším vydáním Axis 1.4, jde o vydání Axis2. Pokud budete hledat Axis 1.5, budete nasměrováni na stránku http://ws.apache.org/axis2/.https://ws.apache.org/axis2/download.cgi Obsahuje seznam verzí produktu Axis2 s popiskem 0.9 až 1.5.1 (včetně matoucí verze 1.4). Verze vydání Axis2, která se má používat pro přenos produktu IBM MQ s protokolem SOAP je 1.4.1. Stáhněte Axis2 1.4.1 ze stránky http://ws.apache.org/axis2/download/1_4_1/download.cgi.

Můžete zvolit generování serverů proxy pro klienty webových služeb pro přenos IBM MQ pro SOAP pomocí produktu **wsimport** nebo nástrojů poskytnutých s prostředím IDE. Eclipse IDE for Java EE Developer 3.5 SR1 používá **wsdl2java**. **wsimport** je dodáván s Java 6. Můžete použít produkt Java 5 ke spouštění serverů proxy klienta generovaných buď s **wsimport** , nebo **wsdl2java**.

Ukázkové klienty webové služby Axis2 dodávané s přenosem produktu IBM MQ pro protokol SOAP byly vyvinuty pomocí produktu **wsimport** ; viz “Ukázka klientů Axis2” na stránce 1291.

Následující úloha demonstruje, jak generovat a používat servery proxy vytvořené pomocí průvodce webovými službami, které jsou zabaleny s produktem Eclipse IDE pro vývojáře Java EE. Ukázkové klienty zobrazují způsob použití serverů proxy vytvořených produktem **wsimport**.

Chcete-li použít průvodce webovými službami, je třeba přidat aplikační server, který podporuje prostředí Axis2 , do pracovní plochy. Tyto kroky ukazují, jak nakonfigurovat profil Liberty pro podporu Axis2 pomocí pracovní plochy.

1. Nakonfigurujte aplikační server používaný v prostředí Eclipse IDE pro produkt Java EE Developers na podporu Axis2. V tomto příkladu nakonfigurujte profil Liberty.
   a. Otevřete předvolby pracovního prostoru a nakonfigurujte server: Otevřete nabídku **Okno** > **Předvolby**.
   b. Zkontrolujte, že nainstalované prostředí JRE je `Java50`: Klepněte na volbu **Instalovaná prostředí JRE**.
   c. Přidejte profil Liberty jako server:
   d. Přidejte Axis2: Klepněte na **Webové služby** > **Axis2 Předvolby**. Na kartě **Axis2 Běhové prostředí** > **Procházet ...** Otevřete adresář obsahující mnoho souborů JAR produktu `Axis2` > **Použít**.
   e. Přidružte Liberty k Axis2: Klepněte na **Webové služby** > **Server a Běhové prostředí**. Pod **`Server`** vyberte **IBM Liberty Server**a pod **`Web service runtime`**vyberte **Apache Axis2** > **Použít** > **OK**
   f. Spusťte server: Otevřete webovou perspektivu a otevřete pohled Servery. Klepněte pravým tlačítkem myši na pohled Servery > **Nový** > **Server**. Volba **IBM Liberty Server** je vybrána a nakonfigurována > **Dokončit**. Spusťte server.
2. Zkontrolujte, zda jste implementovali službu Axis produktu StockQuotedo Liberty, abyste spustili průvodce webovou službou.
3. Chcete-li testovat službu s přenosem IBM MQ pro službu SOAP, implementujte službu do transportu produktu IBM MQ pro modul listener SOAP 1.4, viz profil Liberty.

Informace o této úloze

Produkt Eclipse IDE for Java EE Developers používá produkt Java50 a průvodce webovými službami k vygenerování tříd proxy pro službu. Třídy serveru proxy se liší od tříd vytvořených pomocí nástroje **wsimport** poskytovaného s produktem Java 6. Alternativním přístupem je generování tříd proxy pomocí produktu **wsimport** a import balíků, které vytvoří, do prostředí Eclipse Java EE IDE for Web Developers.

Průvodce webovými službami v prostředí Eclipse IDE for Java EE Developers staví klienta webové služby do webového projektu. Klienta můžete spustit jako jednoduchou aplikaci produktu Java . Nevyžaduje aplikační server. Můžete také přenést kód do projektu produktu Java a nakonfigurovat cestu sestavení tak, aby obsahovala soubory JAR Axis2 .

Postup

1. Vytvořte webový projekt v novém podnikovém projektu:
   a) S ničím vybraným v Průzkumníku projektů > Klepněte pravým tlačítkem myši na prázdný prostor > **Nový** > **Projekt podnikové aplikace** > Název, který je `StockQuoteAxis2EAR` > **Dokončit**. Odpovězte No do okna, které vám dává možnost otevřít perspektivu Java EE.

   Výchozí nastavení je nastaveno pro použití Liberty.

b) Klepněte pravým tlačítkem myši na volby `StockQuoteAxis2EAR` > **Nový** > **Dynamický webový projekt**. Pojmenujte projekt `StockQuoteAxis2WebClient` > Zkontrolujte pole členství EAR a přidejte projekt do **StockQuoteAxis2EAR**. Liberty je vybrán jako cílové běhové prostředí.

c) V sekci Konfigurace na stránce **Nový dynamický webový projekt** > **Upravit ...** > Zkontrolujte fasetu projektu webových služeb Axis2 . **Dynamický webový modul 2.5**, **Java 6.0**a **Liberty** jsou již zaškrtnuty. > **OK** > **Dokončit**. Odpovězte No do okna, které vám dává možnost otevřít perspektivu Java EE.

2. Importujte kód WSDL pro službu do pracovního prostoru a vygenerujte server proxy klienta:

   V tomto příkladu dokument WSDL obsahuje vazbu služby HTTP a stane se cílem pro server proxy statického webového klienta. Adresu URL ve vazbě webové služby můžete upravit tak, aby ukazovala na přenos IBM MQ pro adresu URL SOAP před generováním serveru proxy klienta. Server proxy statického webového klienta je pak služba, která je implementována pro transport SOAP v produktu IBM MQ .

   a) Spusťte průzkumník webových služeb: buď použijte ikonu na řádku s akcemi, nebo **Spustit** > **Spustit průzkumník webových služeb**.

   b) Vyberte průzkumník WSDL klepnutím na ikonu WSDL v okně **Průzkumník webových služeb** > Klepněte na položku **WSDL-hlavní** v okně Navigator > Zadejte adresu URL souboru WSDL StockQuote> **Přejít**.
   V tomto příkladu získáte WSDL přímo ze služby HTTP: `http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis?wsdl`

   c) V Navigatorklepněte na řádek s adresou URL webové služby. V okně **Akce** klepněte na volbu **Importovat WSDL do pracovní plochy** > Vyberte **StockQuoteAxis2WebClient** jako **Projekt pracovní plochy** > Zadejte **Název souboru WSDL**, `StockQuoteAxisHTTP.wsdl` > **Přejít**.

   d) Right-click **StockQuoteAxisHTTP.wsdl** > **Webové služby** > **Generovat klienta**. Zkontrolujte informace o konfiguraci stránky průvodce následujícím způsobem: Server: IBM Liberty Server, webová služba běhového prostředí: Apache Axis2, projekt klienta: StockQuoteAxis2WebClient, projekt klienta EAR: StockQuoteAxisEAR. Chcete-li opravit konfiguraci, klepněte na řádky, které jsou chybné.

   e) Klepněte na tlačítko **Další** > ověřte nastavení generování kódu > **Dokončit**.

   Všimněte si, že nový balík, `soap.server`, je vytvořen a obsahuje proxy, které požadujete.

3. Nakonfigurujte projekt pro spuštění přenosu IBM MQ pro protokol SOAP jako přenos JMS .

   Přenos IBM MQ pro SOAP poskytuje `transportSender`, ale ne `transportReceiver`. Jinými slovy, transport IBM MQ pro SOAP podporuje klienty Axis2 . Momentálně nepodporuje služby Axis2 .

   a) V projektu **StockQuoteAxis2WebClient** klepněte pravým tlačítkem myši na položku `WebContent\WEB-INF\conf\axis2.xml` > **Otevřít pomocí ...** > **Editor XML**.

   b) Vyhledejte poslední `transportSender` (směrem ke konci souboru) a vyhledejte komentář JMS `transportSender` > Right-click řádku > **Přidat před ...** > **transportSender**.

   c) Klepněte pravým tlačítkem myši na **transportSender** > **Add Attribute** > **Name** > Right-click **transportSender** > **Add Attribute** > **Class**.

   d) Klepněte pravým tlačítkem myši na **Název** > **Upravit atribut** > Zadejte **Hodnota:** `jms`

   e) Right-click **Třída** > **Upravit atribut** > Zadejte **Hodnota:** `com.ibm.mq.axis2.transport.jms.WMQJMSTransportSender`. > Uložit.

   f) Přidejte `com.ibm.mq.axis2.transport.jms.WMQJMSTransportSender` do cesty sestavení: Right-click **StockQuoteAxis2WebClient** > **Cesta sestavení** > **Konfigurovat cestu sestavení ...** > Klepněte na kartu **Knihovny** > **Přidat externí soubory JAR ...**. Vyberte všechny soubory JAR v produktu *MQ_INSTALLATION_PATH* `\java\lib` > **OK**.

   MQ_INSTALLATION_PATH je adresář, kde je nainstalován produkt IBM MQ .

4. Vytvořte synchronního statického klienta, otestujte jej pomocí protokolu HTTP a poté převeďte server proxy za účelem spuštění statického klienta pomocí přenosu produktu IBM MQ pro protokol SOAP.

   a) Right-click **Java Prostředky: src** > **Nový** > **Balík** > Název balíku `soap.client` > Dokončit

b) Klepněte pravým tlačítkem myši na volbu **soap.client** > **New** > **Class** > Název třídy `SQA2StaticClient` > **Dokončit**.

c) Nahraďte třídu následujícím kódem a poté klepněte na tlačítko **Uložit**.

```
package soap.client;
import soap.server.StockQuoteAxisServiceStub;
import soap.server.StockQuoteAxisServiceStub.GetQuote;
public class SQA2StaticClient {
  public static void main(String[] args) {
    try {
      StockQuoteAxisServiceStub stub = new StockQuoteAxisServiceStub();
      GetQuote request = new GetQuote();
      request.setSymbol("ibm");
      System.out.println("Response is: "
          + (stub.getQuote(request)).getGetQuoteReturn());
    } catch (Exception e) {
      System.out.println("Exception: " + e.getMessage());
      e.printStackTrace();
    }
  }
}
```

5. Otestujte klienta se službou Axis StockQuoteimplementovanou na server Liberty a s přenosem IBM MQ pro SOAP.

a) V Průzkumníku projektů klepněte pravým tlačítkem myši na položku **SQA2StaticClient** > **Spustit jako ...** > **Java Aplikace**.

Výsledek, `Response is 55.25` , se zobrazí v pohledu Konzola. V pohledu Konzola můžete také vybrat okno konzoly Liberty a zobrazit výstup na serveru Liberty, `StockQuoteAxis called with parameter: ibm` .

b) Server proxy byl sestaven s adresou služby `http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis`, a tak statický klient volá službu spuštěnou na HTTP. Statický klient můžete změnit tak, aby volal službu pomocí transportu produktu IBM MQ pro protokol SOAP. Následující pokyny mění servisní adresu v produktu `StockQuoteAxisServiceStub.java` bez opětovného sestavení serveru proxy a konfigurují běhové parametry produktu `SQA2StaticClient` pro načtení `axis2.xml`. Nakonfigurujete produkt `axis2.xml` pro konfiguraci prostředí Axis2 pro použití transportu produktu IBM MQ pro protokol SOAP.

c) Otevřete `StockQuoteAxisServiceStub.java` >
Nahraďte dva výskyty prvku `http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis` řetězcem,

```
jms:/queue?destination=REQUESTAXIS@QM1
&connectionFactory=()
&initialContextFactory=com.ibm.mq.jms.Nojndi
&targetService=StockQuoteAxis
&replyDestination=SYSTEM.SOAP.RESPONSE.QUEUE
```

d) Spustíte-li nyní produkt `SQA2StaticClient` , bude vyvolána výjimka, protože nebyla nalezena položka `transportSender` nakonfigurovaná pro produkt JMS .
Výjimka:

```
Exception: null java.lang.NullPointerException at
soap.server.StockQuoteAxisServiceStub.getQuote(StockQuoteAxisServiceStub.java:547)
at soap.client.SQA2StaticClient.main(SQA2StaticClient.java:11)
```

e) V Průzkumníku projektů klepněte pravým tlačítkem myši na **SQA2StaticClient** > **Spustit jako ...** > **Spustit konfigurace ...**. Přepněte na kartu **(x) = Argumenty** a do vstupní oblasti **Argumenty VM** zadejte cestu k souboru `axis2.conf` > **Použít** > **Spustit**.
Argument VM je: `-Daxis2.xml=${workspace_loc:StockQuoteAxis2WebClient/WebContent/WEB-INF/conf}/axis2.xml`. Případně můžete zadat standardní cestu ke konfiguračnímu souboru Axis2 .

f) Spusťte příkaz `SQA2StaticClient` znovu. V tomto spuštění používáte přenos IBM MQ pro SOAP. Potvrďte ji tak, že zkontrolujete, že v konzole Liberty není žádný nový výstup. Otevřete konzolu

nebo příkazové okno, které je přidruženo k jednoduchému modulu listenerJava, a výstup je `StockQuoteAxis called with parameter: ibm`.

6. Vytvořte dynamického klienta pro přenos HTTP a IBM MQ pro protokol SOAP a otestujte jej.
   a) Klepněte pravým tlačítkem myši na volbu **soap.client** > **New** > **Class** > Název třídy `SQA2DynamicClient` > **Dokončit**.
   b) Nahraďte třídu následujícím kódem a poté klepněte na tlačítko **Uložit**.

   Obrázek 178. `SQA2DynamicClient.java`

```
package soap.client;
import soap.server.StockQuoteAxisServiceStub;
import soap.server.StockQuoteAxisServiceStub.GetQuote;
public class SQA2DynamicClient {
  public static void main(String[] args) {
    try {
      StockQuoteAxisServiceStub stub = new StockQuoteAxisServiceStub(
          "http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis");
      GetQuote request = new GetQuote();
      request.setSymbol("ibm");
      System.out.println("HTTP Sync: "
          + (stub.getQuote(request)).getGetQuoteReturn());
      stub = new StockQuoteAxisServiceStub(
          "jms:/queue?destination=REQUESTAXIS@QM1"
          + "&connectionFactory=()&initialContextFactory=com.ibm.mq.jms.Nojndi"
          + "&targetService=StockQuoteAxis&replyDestination=SYSTEM.SOAP.RESPONSE.QUEUE");
       System.out.println("JMS sync: "
          + (stub.getQuote(request)).getGetQuoteReturn());
     } catch (Exception e) {
       System.out.println("Exception: " + e.getMessage());
       e.printStackTrace();
     }
   }
 }
```

   c) Vytvořte konfiguraci spuštění pro produkt `SQA2DynamicClient.javaa` přidejte cestu k produktu `axis2.xml`: `-Daxis2.xml=${workspace_loc:StockQuoteAxis2WebClient/WebContent/WEB-INF/conf}/axis2.xml`
   d) Spusťte příkaz `SQA2DynamicClient`. Zkontrolujte výstup konzoly pro produkty `SQA2DynamicClient`, Liberty a **`SimpleJavaListener`**.

7. Vytvořte asynchronního klienta a přistupte k výsledku v obslužné rutině zpětného volání a v hlavním vláknu programu.

   Asynchronní servery proxy klienta vytvořené pomocí průvodce webovou službou pro prostředí Eclipse Java EE IDE for Web Developers se liší od serverů proxy vytvořených produktem **`wsimport`**. Servery proxy produktu **`wsimport`** používají generické typy `Future`, `Response`a `AsyncHandler`.

   Průvodce webové služby pro prostředí Eclipse Java EE IDE for Web Developers vytvoří abstraktní třídu `StockQuoteAxisServiceCallbackHandler`. Je třeba rozšířit produkt `StockQuoteAxisServiceCallbackHandler` a vytvořit obslužnou rutinu zpětného volání.

   a) Klepněte pravým tlačítkem myši na volbu **soap.client** > **New** > **Class** > Název třídy `SQA2CallbackHandler` > **Dokončit**.
   b) Nahraďte třídu následujícím kódem.

```
package soap.client;
import soap.server.StockQuoteAxisServiceCallbackHandler;
import soap.server.StockQuoteAxisServiceStub.GetQuoteResponse;
public class SQA2CallbackHandler
    extends StockQuoteAxisServiceCallbackHandler {
  private boolean complete = false;
  SQA2CallbackHandler() {
    super();
    System.out.println("Callback constructor");
  }
  public void receiveResultgetQuote(GetQuoteResponse response) {
```

```
    System.out.println("Result in Callback " + response.getGetQuoteReturn());
    super.clientData = response;
    complete = true;
  }
  public boolean isComplete() {
    return complete;
  }
}
```

c) Klepněte pravým tlačítkem myši na volbu **soap.client** > **New** > **Class** > Název třídy `SQA2AsyncClient` > **Dokončit**.

d) Nahraďte třídu následujícím kódem.

Obrázek 179. `SQA2AsyncClient.java`

```
package soap.client;
import soap.server.StockQuoteAxisServiceStub;
import soap.server.StockQuoteAxisServiceStub.GetQuote;
import soap.server.StockQuoteAxisServiceStub.GetQuoteResponse;
import soap.server.StockQuoteAxisServiceCallbackHandler;
@SuppressWarnings("unused")
public class SQA2AsyncClient {
  public static void main(String[] args) {
    try {
      StockQuoteAxisServiceStub stub = new StockQuoteAxisServiceStub(
          "http://localhost:8080/StockQuoteAxis/services/StockQuoteAxis");
      GetQuote request = new GetQuote();
      request.setSymbol("ibm");
      System.out.println("HTTP Sync: "
          + (stub.getQuote(request)).getGetQuoteReturn());
      SQA2CallbackHandler callback = new SQA2CallbackHandler();
      stub.startgetQuote(request, callback);
      do {
        System.out.println("Waiting for HTTP callback");
        Thread.sleep(2000);
      } while (!callback.isComplete());
      System.out.println("HTTP poll: "
          + ((GetQuoteResponse) (callback.getClientData()))
                .getGetQuoteReturn());
      stub = new StockQuoteAxisServiceStub(
          "jms:/queue?destination=REQUESTAXIS@QM1"
          + "&connectionFactory=()&initialContextFactory=com.ibm.mq.jms.Nojndi"
          + "&targetService=StockQuoteAxis&replyDestination=SYSTEM.SOAP.RESPONSE.QUEUE");
      System.out.println("JMS Sync: "
          + (stub.getQuote(request)).getGetQuoteReturn());
      callback = new SQA2CallbackHandler();
      stub.startgetQuote(request, callback);
      while (!callback.isComplete()) {
        System.out.println("Waiting for JMS callback");
        Thread.sleep(2000);
      }
      System.out.println("JMS poll: "
        + ((GetQuoteResponse) (callback.getClientData())).getGetQuoteReturn());
    } catch (Exception e) {
        System.out.println("Exception: " + e.getMessage());
        e.printStackTrace();
     }
   }
 }
```

Výstup konzoly je následující:

```
HTTP Sync: 55.25
Callback constructor
Waiting for HTTP callback
Result in Callback 55.25
HTTP poll: 55.25
JMS Sync: 55.25
Callback constructor
Waiting for JMS callback
Result in Callback 55.25
JMS poll: 55.25
```

Ukázka klientů Axis2

Ukázkové servery proxy jsou generovány pomocí nástroje **wsimport** , který je zabalen s produktem Java 6. K dispozici je šest vzorků:

1. DynamicProxyClientSync.java
2. DynamicProxyClientAsyncPolling.java
3. DynamicProxyClientAsyncCallback.java
4. DispatchClientSync.java
5. DispatchClientAsyncPolling.java
6. DispatchClientAsyncCallback.java

Ukázky klienta jsou generovány pro ukázku serveru Axis StockQuote. Vygenerujte WSDL příkazem **amqwdepoyWMQServer** zadáním přepínače -w , abyste vybrali styl `rpcLiteral` . Chcete-li vygenerovat proxy pro ukázky, použijte následující příkaz:

```
wsimport soap.server.StockQuoteAxis_Wmq.wsdl -d generated -keep -p com.ibm.mq.axis2.samples
```

Obrázek 180. DynamicProxyClientSync.java

```
package com.ibm.mq.axis2.samples;

import com.ibm.mq.axis2.samples.proxy.StockQuoteAxis;
import com.ibm.mq.axis2.samples.proxy.StockQuoteAxisService;

public class DynamicProxyClientSync {

  public static void main(String[] args) {
    try {
      System.out.println("Starting sample DynamicProxyClientSync");

      System.out.println("Creating proxy instance for service StockQuoteAxisService");
      StockQuoteAxisService stub = new StockQuoteAxisService();
      StockQuoteAxis service = stub.getSoapServerStockQuoteAxisWmq();

      System.out.println("Invoking getQuoteOneWay OneWay operation synchronously...");
      service.getQuoteOneWay("48");
      System.out.println(" > getQuoteOneWay has returned");

      System.out.println("Invoking getQuote Request Reply operation synchronously...");
      float result = service.getQuote("48");
      System.out.println(" > getQuote has returned result of " + result);

      System.out.println("End of sample");
    }
    catch (Exception fault) {
      // Identify the cause of the Axis Fault
      System.err.println(fault.toString());
      Throwable e = fault.getCause();
      for (int i = 1; e != null; i++) {
        // The toString method on an MQAxisException will cause the message, explanation and
user
        // action.
        System.err.println("Exception(" + i + "): " + e.toString());

        if (e.getCause() != null) {
          e = e.getCause();
        }
        else {
          break;
        }
      } // end of for loop
    } // end of catch block
  }
}
```

Obrázek 181. DynamicProxyClientAsyncPolling.java

```
package com.ibm.mq.axis2.samples;

import java.util.concurrent.CancellationException;

import javax.xml.ws.Response;

import com.ibm.mq.axis2.samples.proxy.StockQuoteAxis;
import com.ibm.mq.axis2.samples.proxy.StockQuoteAxisService;

public class DynamicProxyClientAsyncPolling {

  public static void main(String[] args) {
    try {
      System.out.println("Starting sample DynamicProxyClientAsyncPolling");

      System.out.println("Creating proxy instance for service StockQuoteAxisService");
      StockQuoteAxisService stub = new StockQuoteAxisService();
      StockQuoteAxis service = stub.getSoapServerStockQuoteAxisWmq();

      System.out
          .println("Invoking getQuoteAsync Request Reply operation asynchronously by
polling...");
      Response<Float> response = service.getQuoteAsync("49");

      /** Sleep main thread until response arrives **/
      System.out.println("Waiting for response to arrive...");
      while (!response.isDone()) {
        Thread.sleep(100);
      }
      System.out.println(" > Response received");

      /** Retrieve the result **/
      try {
        Float result = response.get();
        System.out.println(" > getQuoteAsync call has returned result of " + result);
      }
      catch (CancellationException ce) {
        // processing was cancelled via response.cancel()
      }

      System.out.println("End of sample");
    }
    catch (Exception fault) {
      // Identify the cause of the Axis Fault
      System.err.println(fault.toString());
      Throwable e = fault.getCause();
      for (int i = 1; e != null; i++) {
        // The toString method on an MQAxisException will cause the message, explanation and
user
        // action.
        System.err.println("Exception(" + i + "): " + e.toString());

        if (e.getCause() != null) {
          e = e.getCause();
        }
        else {
          break;
        }
      } // end of for loop
    } // end of catch block
  }
}
```

Obrázek 182. DynamicProxyClientAsyncCallback.java

```
package com.ibm.mq.axis2.samples;

import java.util.concurrent.Future;
```

```
import javax.xml.ws.AsyncHandler;
import javax.xml.ws.Response;

import com.ibm.mq.axis2.samples.proxy.StockQuoteAxis;
import com.ibm.mq.axis2.samples.proxy.StockQuoteAxisService;

public class DynamicProxyClientAsyncCallback implements AsyncHandler<Float> {

  public static void main(String[] args) {
    try {
      System.out.println("Starting sample DynamicProxyClientAsyncCallback");

      System.out.println("Creating proxy instance for service StockQuoteAxisService");
      StockQuoteAxisService stub = new StockQuoteAxisService();
      StockQuoteAxis service = stub.getSoapServerStockQuoteAxisWmq();

      DynamicProxyClientAsyncCallback handler = new DynamicProxyClientAsyncCallback();

      System.out
          .println("Invoking getQuoteAsync Request Reply operation asynchronously using a
callback...");
      Future<?> monitor = service.getQuoteAsync("50", handler);
      System.out.println(" > Invoke call has returned");

      /** Sleep main thread until handler has been notified **/
      System.out.println("Waiting for handler to be called...");
      while (!monitor.isDone()) {
        Thread.sleep(100);
      }

      System.out.println("End of sample");
    }
    catch (Exception fault) {
      // Identify the cause of the Axis Fault
      System.err.println(fault.toString());
      Throwable e = fault.getCause();
      for (int i = 1; e != null; i++) {
        // The toString method on an MQAxisException will cause the message, explanation and
user
        // action.
        System.err.println("Exception(" + i + "): " + e.toString());

        if (e.getCause() != null) {
          e = e.getCause();
        }
        else {
          break;
        }
      } // end of for loop
    } // end of catch block
  }

  public void handleResponse(Response<Float> response) {
    try {
      Float result = response.get();
      System.out.println(" > Async Handler has received a result of " + result);
    }
    catch (Exception fault) {
      // Identify the cause of the Axis Fault
      System.err.println("Exception in handleResponce");
      System.err.println(fault.toString());
      Throwable e = fault.getCause();
      for (int i = 1; e != null; i++) {
        // The toString method on an MQAxisException will cause the message, explanation and
user
        // action.
        System.err.println("Exception(" + i + "): " + e.toString());

        if (e.getCause() != null) {
          e = e.getCause();
        }
        else {
          break;
        }
      } // end of for loop
    } // end of catch block
  }

}
```

Obrázek 183. DispatchClientSync.java

```
package com.ibm.mq.axis2.samples;

import javax.xml.namespace.QName;
import javax.xml.soap.MessageFactory;
import javax.xml.soap.SOAPBody;
import javax.xml.soap.SOAPConstants;
import javax.xml.soap.SOAPElement;
import javax.xml.soap.SOAPEnvelope;
import javax.xml.soap.SOAPHeader;
import javax.xml.soap.SOAPMessage;
import javax.xml.soap.SOAPPart;
import javax.xml.ws.Dispatch;
import javax.xml.ws.Service;
import javax.xml.ws.soap.SOAPBinding;

public class DispatchClientSync {

  public static void main(String[] args) {
    try {
      System.out.println("Starting sample DispatchClientSync");

      String endpointUrl = "jms:/queue?destination=SOAPJ.demos@WMQSOAP.DEMO.QM&"
          + "connectionFactory=connectQueueManager(WMQSOAP.DEMO.QM)"
          +
"&initialContextFactory=com.ibm.mq.jms.Nojndi&targetService=soap.server.StockQuoteAxis.java";

      QName serviceName = new QName("soap.server.StockQuoteAxis_Wmq", "StockQuoteAxisService");
      QName portName = new QName("soap.server.StockQuoteAxis_Wmq",
"soap.server.StockQuoteAxis_Wmq");

      Service service = Service.create(serviceName);
      service.addPort(portName, SOAPBinding.SOAP11HTTP_BINDING, endpointUrl);

      /** Create a Dispatch instance from a service **/
      System.out.println("Creating dispatch instance for service StockQuoteAxisService");
      Dispatch<SOAPMessage> dispatch = service.createDispatch(portName, SOAPMessage.class,
          Service.Mode.MESSAGE);
      System.out.println(" > Dispatch instance created.");

      /************************************
       * Create OneWay SOAPMessage request.
       ************************************/
      MessageFactory mf = MessageFactory.newInstance(SOAPConstants.SOAP_1_1_PROTOCOL);

      System.out.println("\nCreating a OneWay SOAP Message");
      SOAPMessage request = mf.createMessage();

      /** Obtain the SOAPEnvelope and header and body elements **/
      SOAPPart part = request.getSOAPPart();
      SOAPEnvelope env = part.getEnvelope();
      SOAPHeader header = env.getHeader();
      SOAPBody body = env.getBody();

      /** Construct the message payload **/
      SOAPElement operation = body.addChildElement("getQuoteOneWay", "ns1",
          "soap.server.StockQuoteAxis_Wmq");
      SOAPElement value = operation.addChildElement("in0");
      value.addAttribute(new QName("https://www.w3.org/2001/XMLSchema-instance", "type"),
"string");
      value.addTextNode("XXX");
      request.saveChanges();
      System.out.println(" > SOAP Message created.");

      /** Invoke the service endpoint **/
      System.out.println("Invoking getQuoteOneWay OneWay operation synchronously...");
      dispatch.invokeOneWay(request);
      System.out.println(" > getQuoteOneWay call has returned");

      /****************************************
       * Create Request Reply SOAPMessage request.
       ****************************************/
      mf = MessageFactory.newInstance(SOAPConstants.SOAP_1_1_PROTOCOL);

      System.out.println("\nCreating a Request Reply SOAP Message");
```

```
      request = mf.createMessage();

      /** Obtain the SOAPEnvelope and header and body elements **/
      part = request.getSOAPPart();
      env = part.getEnvelope();
      header = env.getHeader();
      body = env.getBody();

      /** Construct the message payload **/
      operation = body.addChildElement("getQuote", "ns1", "soap.server.StockQuoteAxis_Wmq");
      value = operation.addChildElement("in0");
      value.addAttribute(new QName("https://www.w3.org/2001/XMLSchema-instance", "type"),
"string");
      value.addTextNode("XXX");
      request.saveChanges();
      System.out.println(" > SOAP Message created.");

      /** Invoke the service endpoint **/
      System.out.println("Invoking getQuote Request Reply operation synchronously...");
      SOAPMessage ans = dispatch.invoke(request);
      System.out.println(" > getQuote call has returned");

      /** Retrieve the result **/
      part = ans.getSOAPPart();
      env = part.getEnvelope();
      body = env.getBody();

      /** Define name of the SOAP folders we are interested in **/
      QName responseName = new QName("soap.server.StockQuoteAxis_Wmq", "getQuoteResponse");
      QName resultName = new QName("getQuoteReturn");

      /** Retrieve result from SOAP envelope **/
      System.out.println("Parsing SOAP response...");
      SOAPElement bodyElement = (SOAPElement) body.getChildElements(responseName).next();
      SOAPElement responseElement = (SOAPElement)
bodyElement.getChildElements(resultName).next();
      String message = responseElement.getValue();
      System.out.println(" > Response contains result of " + message);

      System.out.println("End of sample");
    }
    catch (Exception fault) {
      // Identify the cause of the Axis Fault
      System.err.println(fault.toString());
      Throwable e = fault.getCause();
      for (int i = 1; e != null; i++) {
        // The toString method on an MQAxisException will cause the message, explanation and
user
        // action.
        System.err.println("Exception(" + i + "): " + e.toString());

        if (e.getCause() != null) {
          e = e.getCause();
        }
        else {
          break;
        }
      } // end of for loop
    } // end of catch block
  }
}
```

Obrázek 184. DispatchClientAsyncPolling.java

```
package com.ibm.mq.axis2.samples;

import javax.xml.namespace.QName;
import javax.xml.soap.MessageFactory;
import javax.xml.soap.SOAPBody;
import javax.xml.soap.SOAPConstants;
import javax.xml.soap.SOAPElement;
import javax.xml.soap.SOAPEnvelope;
import javax.xml.soap.SOAPHeader;
import javax.xml.soap.SOAPMessage;
import javax.xml.soap.SOAPPart;
import javax.xml.ws.Dispatch;
```

```
import javax.xml.ws.Response;
import javax.xml.ws.Service;
import javax.xml.ws.soap.SOAPBinding;

public class DispatchClientAsyncPolling {

  public static void main(String[] args) {
    try {
      System.out.println("Starting sample DispatchClientAsyncPolling");

      String endpointUrl = "jms:/queue?destination=SOAPJ.demos@WMQSOAP.DEMO.QM&"
          + "connectionFactory=connectQueueManager(WMQSOAP.DEMO.QM)"
          +
"&initialContextFactory=com.ibm.mq.jms.Nojndi&targetService=soap.server.StockQuoteAxis.java";

      QName serviceName = new QName("soap.server.StockQuoteAxis_Wmq", "StockQuoteAxisService");
      QName portName = new QName("soap.server.StockQuoteAxis_Wmq",
"soap.server.StockQuoteAxis_Wmq");

      Service service = Service.create(serviceName);
      service.addPort(portName, SOAPBinding.SOAP11HTTP_BINDING, endpointUrl);

      /** Create a Dispatch instance from a service.* */
      System.out.println("Creating dispatch instance for service StockQuoteAxisService");
      Dispatch<SOAPMessage> dispatch = service.createDispatch(portName, SOAPMessage.class,
          Service.Mode.MESSAGE);
      System.out.println(" > Dispatch instance created.");

      /** Create SOAPMessage request. * */
      MessageFactory mf = MessageFactory.newInstance(SOAPConstants.SOAP_1_1_PROTOCOL);

      System.out.println("Creating a Request Reply SOAP Message");
      SOAPMessage request = mf.createMessage();

      /** Obtain the SOAPEnvelope and header and body elements **/
      SOAPPart part = request.getSOAPPart();
      SOAPEnvelope env = part.getEnvelope();
      SOAPHeader header = env.getHeader();
      SOAPBody body = env.getBody();

      /** Construct the message payload **/
      SOAPElement operation = body.addChildElement("getQuote", "ns1",
          "soap.server.StockQuoteAxis_Wmq");
      SOAPElement value = operation.addChildElement("in0");
      value.addAttribute(new QName("https://www.w3.org/2001/XMLSchema-instance", "type"),
"string");
      value.addTextNode("XXX");
      request.saveChanges();
      System.out.println(" > SOAP Message created.");

      /** Invoke the service endpoint **/
      System.out.println("Invoking getQuote Request Reply operation asynchronously by
polling...");
      Response<SOAPMessage> response = dispatch.invokeAsync(request);
      System.out.println(" > getQuote call has returned");

      /** Sleep main thread until response arrives **/
      System.out.println("Waiting for response to arrive...");
      while (!response.isDone()) {
        Thread.sleep(100);
      }
      System.out.println(" > Response received");

      /** retrieve the result **/
      SOAPMessage ans = response.get();
      part = ans.getSOAPPart();
      env = part.getEnvelope();
      body = env.getBody();

      /** Define name of the SOAP folders we are interested in **/
      QName responseName = new QName("soap.server.StockQuoteAxis_Wmq", "getQuoteResponse");
      QName resultName = new QName("getQuoteReturn");

      /** Retrieve result from SOAP envelope **/
      SOAPElement bodyElement = (SOAPElement) body.getChildElements(responseName).next();
      SOAPElement responseElement = (SOAPElement)
bodyElement.getChildElements(resultName).next();
      String message = responseElement.getValue();
      System.out.println(" > Response contains result of " + message);

      System.out.println("End of sample");
```

```
    }
    catch (Exception fault) {
      // Identify the cause of the Axis Fault
      System.err.println(fault.toString());
      Throwable e = fault.getCause();
      for (int i = 1; e != null; i++) {
        // The toString method on an MQAxisException will cause the message, explanation and
user
        // action.
        System.err.println("Exception(" + i + "): " + e.toString());

        if (e.getCause() != null) {
          e = e.getCause();
        }
        else {
          break;
        }
      } // end of for loop
    } // end of catch block
  }
}
```

Obrázek 185. DispatchClientAsyncCallback.java

```
package com.ibm.mq.axis2.samples;

import java.util.concurrent.Future;

import javax.xml.namespace.QName;
import javax.xml.soap.MessageFactory;
import javax.xml.soap.SOAPBody;
import javax.xml.soap.SOAPConstants;
import javax.xml.soap.SOAPElement;
import javax.xml.soap.SOAPEnvelope;
import javax.xml.soap.SOAPHeader;
import javax.xml.soap.SOAPMessage;
import javax.xml.soap.SOAPPart;
import javax.xml.ws.AsyncHandler;
import javax.xml.ws.Dispatch;
import javax.xml.ws.Response;
import javax.xml.ws.Service;
import javax.xml.ws.soap.SOAPBinding;

public class DispatchClientAsyncCallback implements AsyncHandler<SOAPMessage> {

  public static void main(String[] args) {
    try {
      System.out.println("Starting sample DispatchClientAsyncCallback");

      String endpointUrl = "jms:/queue?destination=SOAPJ.demos@WMQSOAP.DEMO.QM&"
          + "connectionFactory=connectQueueManager(WMQSOAP.DEMO.QM)"
          +
"&initialContextFactory=com.ibm.mq.jms.Nojndi&targetService=soap.server.StockQuoteAxis.java";

      QName serviceName = new QName("soap.server.StockQuoteAxis_Wmq", "StockQuoteAxisService");
      QName portName = new QName("soap.server.StockQuoteAxis_Wmq",
"soap.server.StockQuoteAxis_Wmq");

      Service service = Service.create(serviceName);
      service.addPort(portName, SOAPBinding.SOAP11HTTP_BINDING, endpointUrl);

      /** Create a Dispatch instance from a service.* */
      System.out.println("Creating dispatch instance for service StockQuoteAxisService");
      Dispatch<SOAPMessage> dispatch = service.createDispatch(portName, SOAPMessage.class,
          Service.Mode.MESSAGE);
      System.out.println(" > Dispatch instance created.");

      /** Create SOAPMessage request. * */
      MessageFactory mf = MessageFactory.newInstance(SOAPConstants.SOAP_1_1_PROTOCOL);

      System.out.println("Creating a Request Reply SOAP Message");
      SOAPMessage request = mf.createMessage();

      /** Obtain the SOAPEnvelope and header and body elements **/
      SOAPPart part = request.getSOAPPart();
      SOAPEnvelope env = part.getEnvelope();
```

```
      SOAPHeader header = env.getHeader();
      SOAPBody body = env.getBody();

      /** Construct the message payload. **/
      SOAPElement operation = body.addChildElement("getQuote", "ns1",
          "soap.server.StockQuoteAxis_Wmq");
      SOAPElement value = operation.addChildElement("in0");
      value.addAttribute(new QName("https://www.w3.org/2001/XMLSchema-instance", "type"),
"string");
      value.addTextNode("XXX");
      request.saveChanges();
      System.out.println(" > SOAP Message created.");

      /** Invoke the service endpoint. **/
      DispatchClientAsyncCallback handler = new DispatchClientAsyncCallback();

      System.out
          .println("Invoking getQuote Request Reply operation asynchronously using a
callback...");
      Future<?> monitor = dispatch.invokeAsync(request, handler);
      System.out.println(" > getQuote call has returned");

      /** Sleep main thread until handler has been notified **/
      System.out.println("Waiting for handler to be called...");
      while (!monitor.isDone()) {
        Thread.sleep(100);
      }

      System.out.println("End of sample");
    }
    catch (Exception fault) {
      // Identify the cause of the Axis Fault
      System.err.println(fault.toString());
      Throwable e = fault.getCause();
      for (int i = 1; e != null; i++) {
        // The toString method on an MQAxisException will cause the message, explanation and
user
        // action.
        System.err.println("Exception(" + i + "): " + e.toString());

        if (e.getCause() != null) {
          e = e.getCause();
        }
        else {
          break;
        }
      } // end of for loop
    } // end of catch block
  }

  public void handleResponse(Response<SOAPMessage> response) {
    try {
      // retrieve the result
      SOAPMessage ans = response.get();
      SOAPPart part = ans.getSOAPPart();
      SOAPEnvelope env = part.getEnvelope();
      SOAPBody body = env.getBody();

      /** Define name of the SOAP folders we are interested in **/
      QName responseName = new QName("soap.server.StockQuoteAxis_Wmq", "getQuoteResponse");
      QName resultName = new QName("getQuoteReturn");

      /** Retrieve result from SOAP envelope **/
      SOAPElement bodyElement = (SOAPElement) body.getChildElements(responseName).next();
      SOAPElement responseElement = (SOAPElement)
bodyElement.getChildElements(resultName).next();
      String result = responseElement.getValue();

      System.out.println(" > Async Handler has received a result of " + result);
    }
    catch (Exception fault) {
      // Identify the cause of the Axis Fault
      System.err.println("Exception in handleResponce");
      System.err.println(fault.toString());
      Throwable e = fault.getCause();
      for (int i = 1; e != null; i++) {
        // The toString method on an MQAxisException will cause the message, explanation and
user
        // action.
        System.err.println("Exception(" + i + "): " + e.toString());

        if (e.getCause() != null) {
```

```
          e = e.getCause();
        }
        else {
          break;
        }
      } // end of for loop
    } // end of catch block
  }
}
```

Související úlohy

Vývoj klienta JAX-RPC pro produkt WebSphere transport pro protokol SOAP pomocí prostředí Eclipse
Vyviňte klienta webové služby Axis 1.4 , který má být spuštěn pomocí přenosu produktu IBM MQ pro protokol SOAP.

Vyvíjení klienta .NET 1 nebo 2 pro přenos WebSphere pro protokol SOAP pomocí produktu Microsoft Visual Studio 2012
Vytvořte klienta webové služby .NET 1 nebo 2, který má být spuštěn pomocí přenosu produktu IBM MQ pro protokol SOAP.

Vyvíjení klienta .NET 1 nebo 2 pro přenos WebSphere pro protokol SOAP pomocí produktu Microsoft Visual Studio 2012

Vytvořte klienta webové služby .NET 1 nebo 2, který má být spuštěn pomocí přenosu produktu IBM MQ pro protokol SOAP.

Než začnete

Důležité: .NET Framework 2.0 je předpokladem pro tuto úlohu, proto se ujistěte, že jste jej nainstalovali dříve, než začnete.

Vývoj klienta produktu .NET 1 nebo 2 lze spustit několika různými způsoby:

1. Pomocí produktu **amqwdeployWMQService** vygenerujte stuby klienta z webové služby a importujte je do produktu Microsoft Visual Studio.
2. Použijte **java2wsdl** ke generování WSDL z implementace webové služby Java a potom použijte `wsdl.exe`, který je dodáván s produktem .NETke generování stubů klienta.
3. Vygenerujte WSDL z implementace služby .NET `.asmx` služby pomocí produktu **amqswsdl**a poté použijte příkaz `wsdl.exe`.
4. Pokud jste vyvinuli a implementovali službu pro HTTP, použijte odkaz **Přidat webový odkaz ...** pomocí průvodce v produktu Microsoft Visual Studio nakonfigurujte klienta pro přístup ke službě HTTP. Upravte adresu URL tak, aby odkazovala na službu implementovanou v produktu IBM MQ pro protokol SOAP.

Úloha používá službu vyvinutou v produktu “Vývoj služby .NET 1 nebo 2 pro přenos IBM MQ pro SOAP pomocí produktu Microsoft Visual Studio 2012” na stránce 1270.

Informace o této úloze

Postupujte takto, chcete-li vytvořit přenos klienta .NET 1 nebo 2 pro protokol HTTP a IBM MQ pro protokol SOAP.

Postup

1. Vytvořte aplikaci konzoly klienta a upravte ji tak, aby vyvolala webovou službu HTTP StockQuote .
   a) Klepněte pravým tlačítkem myši na volbu **Řešení 'StockQuoteDotNet'** v **Průzkumníku řešení** > Přidat ... > Nový projekt. Vyberte typ projektu **C#** , **.NET Framework 2.0**a **Aplikace konzoly**. Pojmenujte projekt `StockQuoteClientDotNet` > **OK**
   b) Klepněte pravým tlačítkem myši na volbu **Řešení 'StockQuoteDotNet'** v **Průzkumníku řešení** > Přidat ... > Nový projekt. Vyberte typ projektu **C#** , **.NET Framework 2.0**a **Aplikace konzoly**. Pojmenujte projekt `StockQuoteClientDotNet` > **OK**
   c) Right-click **StockQuoteClientDotNet** > **Nastavit jako projekt spuštění**.

d) Right-click **StockQuoteClientDotNet** > **Přidat webový odkaz ...** > Procházet a hledat webové služby v tomto řešení > Vyberte **StockQuote** > **Přidat odkaz**. Všimněte si, že jste přidali webový odkaz na lokálního hostitele a nový konfigurační soubor `app.config`.

e) V průzkumníku řešení změňte název aplikace konzoly z `Program.cs` na `StockQuoteClientDotNet.cs` > Klepnutím na tlačítko **OK** změníte všechna použití `Program.cs` na `StockQuoteClientDotNet.cs`.

f) Nahraďte obsah souboru `StockQuoteClientDotNet.cs` kódem v souboruObrázek 186 na stránce 1300.

```
using System;
using StockQuoteClientDotNet.localhost;
namespace StockQuoteClientDotNet {
  class StockQuoteClientDotNet {
    static void Main(string[] args) {
      try {
        StockQuote stockobj = new StockQuote();
        Console.WriteLine("http reply is: "
          + stockobj.getNonInlineQuote("http request");
      }
      catch (System.Exception e) {
        Console.WriteLine("Exception thrown: " + e.ToString());
      }
      Console.ReadLine();
    }
  }
}
```

Obrázek 186. Obslužný program HTTP StockQuoteClientDotNet

g) Spusťte produkt StockQuoteClientDotNet na test vůči službě `StockQuote.asmx` :

i) Stiskněte tlačítko **F5**, klepněte na zelenou šipku v řádku s akcemi nebo na tlačítko **Ladit** > **Spustit ladění (F5)**.

Pokud se projekt StockQuoteDOtNet nachází ve stejném řešení, spustí se automaticky. V opačném případě je třeba nejprve spustit službu.

Příkazové okno s výsledky se otevře za pracovním prostorem. Příkaz `Console.ReadLine();` zabrání, aby se zavřel, dokud nestisknete klávesu **Enter**.

Tip: Ujistěte se, že `StockQuote.asmx` je počáteční stránka v projektu `StockQuoteDotNet` .

2. Upravte soubor StockQuoteClientDotNet tak, aby volal službu `StockQuote.asmx` pomocí přenosu IBM MQ pro SOAP.

a) Přidejte řádky zobrazené tučným písmem na klienta.

```
using System;
using StockQuoteClientDotNet.localhost;
namespace StockQuoteClientDotNet {
  class StockQuoteClientDotNet {
    static void Main(string[] args) {
      try {
        IBM.WMQSOAP.Register.Extension();
        StockQuote stockobj = new StockQuote();
        Console.WriteLine("http reply is: "
          + stockobj.getNonInlineQuote("http request");
        stockobj.Url = "jms:/queue?"
          + "initialContextFactory=com.ibm.mq.jms.Nojndi"
          + "&connectionFactory=()&destination=REQUESTDOTNET@QM1"
          + "&targetService=StockQuote.asmx";
        Console.WriteLine("jms reply is: "
          + stockobj.getNonInlineQuote("jms request"));
      }
      catch (System.Exception e) {
        Console.WriteLine("Exception thrown: " + e.ToString());
      }
      Console.ReadLine();
    }
  }
}
```

Obrázek 187. Upravený program StockQuoteClientDotNet

Případně upravte výchozí adresu URL. Otevřete nabídku **StockQuoteClientDotNet** > **Properties** > **Settings.settings** a změňte hodnotu vlastnosti `StockQuoteClientDotNet_localhost_StockQuote` na přenos IBM MQ pro adresu URL protokolu SOAP.

b) Přidat odkaz na `amqsoap.dll`

i) V projektu **StockQuoteClientDotNet** v **Průzkumníku řešení**klepněte pravým tlačítkem myši na položku **Odkazy** > **Přidat odkaz ...** > Klepněte na kartu **Procházet** > přejděte na nabídku *MQ_INSTALLATION_PATH*\bin > Vyberte volbu **amqsoap.dll** > **OK**. *MQ_INSTALLATION_PATH* je adresář, kde je nainstalován produkt IBM MQ .

3. Otestujte klienta se službou `StockQuote.asmx` pomocí přenosu IBM MQ pro SOAP.

a) Otevřete příkazové okno v adresáři projektu StockQuoteDotNet : `.\StockQuoteDotNet\StockQuoteDotNet` > Ověřte, že `.bin\StockQuoteDotNet.dll` existuje. Pokud tomu tak není, znovu sestavte řešení.

b) Napište příkaz **amqwRegisterdotNet**.

amqwRegisterdotNet je třeba spustit pouze jednou při instalaci.

c) Pokud jste spustili příkaz **amqwdeployWMQServer** s parametrem `genAsmxWMQBits`, spusťte modul listener SOAP produktu .NET : `generated\server\startWMQNListener`

d) Případně spusťte modul listener přímo:

```
amqwSOAPNETListener -u "jms:/queue?
destination=REQUESTDOTNET@QM1
&connectionFactory=()&initialContextFactory=com.ibm.mq.jms.Nojndi
&targetService=StockQuote.asmx&replyDestination=SYSTEM.SOAP.RESPONSE.QUEUE"
-w C:\IBM\ID\StockQuoteDotNet\StockQuoteDotNet -n 10
```

4. V produktu Visual Studio 2012 stisknutím klávesy **F5** spusťte příkaz StockQuoteClientDotNet.

Klienti webové služby .NET Framework 1 a .NET Framework 2

Ukázkový klient produktu .NET poskytovaný s přenosem produktu IBM MQ pro SOAP používá vygenerované stuby pro volání ukázkových služeb Axis a .NET .

V případě klientů .NET Framework 1 a .NET Framework 2 poskytuje produkt IBM MQ přístup k webovým službám pomocí klientů produktu .NET . Příkaz **amqwdeployWMQService** má volbu genProxiestoDotNet, která generuje stuby klienta .NET Framework 1 nebo .NET Framework 2 pro webovou službu. Můžete také použít stuby klienta generované nástrojem .NET **wsdl** nebo produktem Microsoft Visual Studio 2012.

Ukázkové klienty webové služby .NET Framework 1 a .NET jsou nainstalovány v produktu *MQ_INSTALLATION_PATH* \tools\soap\samples\dotnet. *MQ_INSTALLATION_PATH* je adresář, kde je nainstalován produkt IBM MQ .

SQVB2Axis.vb
SQVB2Axis.vb,Obrázek 188 na stránce 1302, je klient Visual Basic k volání služby **StockQuoteAxisService** .

SQVB2DotNet.vb
QVB2DotNet.vb,Obrázek 189 na stránce 1302, je klient Visual Basic k volání služby **StockQuoteDotNet** .

SQCS2Axis.cs
SQCS2Axis.cs,Obrázek 190 na stránce 1303, je klient C# pro volání služby **StockQuoteAxisService** . Adresu URL služby můžete přepsat zadáním adresy URL na příkazovém řádku.

SQCS2DotNet.cs
SQCS2DotNet.cs,Obrázek 191 na stránce 1303, je klient C# pro volání služby **StockQuoteDotNet** . Adresu URL služby můžete přepsat zadáním adresy URL na příkazovém řádku.

```
Module SQVB2Axis
   Function Main(ByVal CmdArgs() As String) As Integer
      IBM.WMQSOAP.Register.Extension()
      Dim obj As New StockQuoteAxisService()
      Dim res As Single = obj.getQuote("fromcs")
      System.Console.WriteLine("SQVB2Axis: reply is: '{0}'", res)
   End Function
End Module
```

Obrázek 188. SQVB2Axis

```
Module SQVB2DotNet
   Function Main(ByVal CmdArgs() As String) As Integer
      IBM.WMQSOAP.Register.Extension()
      Dim obj as new StockQuoteDotNet()
      Dim res as Single = obj.getQuote("fromcs")
      System.Console.WriteLine("SQVB2DotNet: reply is: '{0}'", res)
   End Function
End Module
```

Obrázek 189. SQVB2DotNet

```
using System;
class SQCS2Axis {
   [STAThread]
   static void Main(string[] args) {
      try {
         IBM.WMQSOAP.Register.Extension();
         StockQuoteAxisService stockobj = new StockQuoteAxisService();
         if (args.GetLength(0) >= 1)
            stockobj.Url = args[0];
         System.Single res = stockobj.getQuote("XXX");
         Console.WriteLine("SQCS2Axis RPC reply is: " + res);
      }
      catch (System.Exception e) {
         Console.WriteLine("\n>>> EXCEPTION WHILE RUNNING SQCS2Axis DEMO <<<\n"
            + e.ToString());
      }
   }
}
```

Obrázek 190. SQCS2Axis

```
using System;
class SQCS2DotNet {
   [STAThread]
   static void Main(string[] args) {
      try {
         IBM.WMQSOAP.Register.Extension();
         StockQuoteDotNet stockobj = new StockQuoteDotNet();
         if (args.GetLength(0) >= 1)
            stockobj.Url = args[0];
         System.Single res = stockobj.getQuote("XXX");
         Console.WriteLine("RPC reply is: " + res);
         if (args.GetLength(0) == 0) {
            res = stockobj.getQuoteDOC("XXX");
            Console.WriteLine("DOC reply is: " + res);
         }
      }
      catch (System.Exception e) {
         Console.WriteLine("\n>>> EXCEPTION WHILE RUNNING SQCS2DotNet DEMO <<<\n"
            + e.ToString());
      }
   }
}
```

Obrázek 191. SQCS2DotNet

Související úlohy

Vývoj klienta JAX-RPC pro produkt WebSphere transport pro protokol SOAP pomocí prostředí Eclipse
Vyviňte klienta webové služby Axis 1.4 , který má být spuštěn pomocí přenosu produktu IBM MQ pro protokol SOAP.

Vyvíjení klienta JAX-WS pro produkt WebSphere transport pro protokol SOAP s použitím prostředí Eclipse
Vytvořte klienta webové služby Axis2 , který má být spuštěn pomocí přenosu produktu IBM MQ pro protokol SOAP. Ukázkový klient Axis2 dodávaný s přenosem IBM MQ pro protokol SOAP je uveden v seznamu a příkaz **wsimport** použitý ke generování serverů proxy.

Implementace webových služeb pomocí přenosu produktu IBM MQ pro protokol SOAP

Nasaďte webovou službu na jeden z mnoha různých serverových prostředí a připojte se k němu pomocí přenosu IBM MQ pro SOAP.

Než začnete

Vytvořte webovou službu a otestujte ji pomocí protokolu SOAP prostřednictvím protokolu HTTP v cílovém prostředí.

Informace o této úloze

Můžete implementovat webovou službu pro provoz s přenosem IBM MQ pro protokol SOAP v řadě různých běhových prostředí SOAP. Službu můžete implementovat na Axis 1.4 pouze s použitím softwaru instalovaného s produktem IBM MQ. V případě jiných běhových prostředí musíte instalovat další software.

Nejste omezeni na spuštění přenosu IBM MQ pro protokol SOAP na servery, pro které jsou pokyny k implementaci. Postupujte podle pokynů pro implementaci služby do jednoho z uvedených prostředí.

Poznámka: Některá integrovaná prostředí nabízejí protokol SOAP prostřednictvím produktu JMS s doporučenou vazbou JMS protokolu SOAP W3C a také přenosem IBM MQ pro vazbu SOAP. Vydání produktu IBM MQ, včetně 7.0.1.2, podporuje pouze přenos IBM MQ pro vazbu SOAP. Počínaje verzí 7.0.1.3 můžete implementovat klienty Axis2 pomocí identifikátoru URI, který odpovídá doporučenému doporučení W3C pro protokol SOAP prostřednictvím produktu JMS. Viz výukový program Vývoj aplikací webových služeb SOAP/JMS JAX-WS s WebSphere Application Server V7 a Rational Application Developer 7.5.

Implementace služby na Axis 1.4 k použití pro přenos WebSphere pro SOAP pomocí produktu `amqwdeployWMQService`

Implementujte službu Axis 1.4 pro transport SOAP pro produkt IBM MQ vytvořením adresáře implementace, spuštěním příkazu **amqwdeployWMQService** a spuštěním modulu listener Axis 1.4 .

Než začnete

1. Postupujte podle pokynů pro instalaci přenosu produktu IBM MQ pro protokol SOAP
2. Ověřte instalaci a prostředí pomocí příkazu **runivt** .
3. Chcete-li znovu nasadit službu:
   a. Odstraňte podadresář `./generated` a všechny jeho podadresáře.
   b. Odeberte požadavky z cílové fronty a odstraňte ji.
   c. Pokračujte podle pokynů z kroku “2” na stránce 1304.

Informace o této úloze

Tyto pokyny slouží k první implementaci služby Axis 1.4 . Chcete-li restartovat službu Axis 1.4 , spusťte znovu modul listener protokolu SOAP 1.4 : krok “11” na stránce 1305.

Chcete-li implementovat novou službu Axis 1.4 do transportu produktu IBM MQ pro protokol SOAP, postupujte podle následujících pokynů:

Postup

1. Vytvořte adresář *`deployDir`* , který bude obsahovat soubory implementace.

   Obslužný program implementace vyžaduje, aby byla každá služba implementována ze samostatného adresáře.
2. Otevřete příkazové okno na systému Windowsnebo příkazový shell pomocí systému X Window System na systémech UNIX and Linux , v *`deployDir`* ke spuštění **amqwdeployWMQService**.
3. Spuštěním příkazu **amqwsetcp** nastavte cestu ke třídě.

   Prostředí JRE a sada JDK musí být v cestě ke třídě, ve verzi 5.0 nebo novější a na stejné úrovni verze.
4. Zkopírujte zdroj třídy *`className`*`.javado` *`deployDir`* .
5. Zkopírujte všechny zdrojové soubory Java ve stejném balíku jako *className* do *`deployDir`*`/`*packageName*, kde *packageName* je adresářový strom odpovídající názvu balíku.

6. Spustit **javac** *packageName*. *className*.

   Možná budete muset přidat cestu k aktuálnímu adresáři " . ", nebo do adresáře *packageName* pro **javac** , abyste našli ostatní třídy.
7. Vytvořit WSDL osy pro službu:

```
amqwdeployWMQService -f packageName.className.java -c genAxisWsdl
-v -u "jms:/queue?destination=queueName
&initialContextFactory=com.ibm.mq.jms.Nojndi
&connectionFactory=(connectQueueManager(QmgrName)binding(auto))"
```

8. Vytvořte prostředky IBM MQ pro službu:

```
amqwdeployWMQService -f packageName.className.java -c genAxisWMQBits
-v -u  "jms:/queue?destination=queueName
&initialContextFactory=com.ibm.mq.jms.Nojndi
&connectionFactory=(connectQueueManager(QmgrName)binding(auto))"
```

   Tip:

   Chcete-li nastavit nového správce front a prostředky, které potřebuje, abyste mohli provádět vývoj a testování, spusťte produkt **setupWMQSOAP**.

   Chcete-li nastavit nového správce front jako výchozí, vezměte kopii produktu **setupWMQSOAP** z adresáře *WMQ installation directory*\tools\soap\samples a do řádku přidejte parametr -q .

```
call :try -q crtmqm %QMGR%
```

9. Vytvořte modul listener Axis a implementujte službu:

```
amqwdeployWMQService -f packageName.className.java -c AxisDeploy
-v -u "jms:/queue?destination=queueName
&initialContextFactory=com.ibm.mq.jms.Nojndi
&connectionFactory=(connectQueueManager(QmgrName)binding(auto))"
```

10. Pokud potřebujete generovat kód WSDL pro službu, generovat stuby klienta nebo servery proxy klienta, spusťte **amqwdeployWMQService** s jedním z následujících parametrů:

    - genAsmxWsdl
    - genAxisWsdl
    - genProxiesToDotNet
    - genProxiestoOsa

    Poznámka: Před generováním serverů proxy je třeba vygenerovat kód WSDL. Volba AllAxis selže, pokud není proměnná CLASSPATH nastavena pro nalezení všech tříd, které jsou importovány ke kompilaci *className*.java. Existuje-li více souborů Java v balíku obsahujícím *className*.java, musíte je nejprve zkompilovat pomocí produktu **javac**. **amqwdeployWMQService** -f *packageName*. *className*.java -c CompileJava kompiluje pouze *className*.java.
11. Spusťte generovaný modul listener Axis.

```
.\generated\server\startWMQJListener.cmd
```

Související úlohy

Implementace služby pro službu .NET Framework 1 nebo 2 pro použití přenosu produktu IBM MQ pro protokol SOAP

Implementujte službu .NET Framework 1 nebo 2 do transportu produktu IBM MQ pro protokol SOAP. Vytvořte adresář implementace, spusťte příkaz **amqwdeployWMQService** a spusťte modul listener produktu .NET .

Implementace služby pro server CICS Transaction Server pro použití přenosu WebSphere pro SOAP

Přenos produktu IBM MQ pro SOAP je integrován do podpory webových služeb produktu CICS Transaction Server 4.1 .

Implementace služby do produktu WebSphere Application Server pro použití přenosu WebSphere pro protokol SOAP
Přenos produktu IBM MQ pro SOAP je integrován do sběrnice pro integraci služeb v produktu WebSphere Application Server.

Implementace služby pro koncový bod služby produktu WebSphere ESB a Process Server pro použití přenosu produktu WebSphere pro protokol SOAP
Přenos produktu IBM MQ pro SOAP není přímo podporován produkty WebSphere ESB a Process Server. Musíte nakonfigurovat vlastní export.

Implementace služby pro službu .NET Framework 1 nebo 2 pro použití přenosu produktu IBM MQ pro protokol SOAP

Implementujte službu .NET Framework 1 nebo 2 do transportu produktu IBM MQ pro protokol SOAP. Vytvořte adresář implementace, spusťte příkaz **amqwdeployWMQService** a spusťte modul listener produktu .NET .

Než začnete

1. Postupujte podle pokynů pro instalaci přenosu produktu IBM MQ pro protokol SOAP
2. Ověřte instalaci a prostředí pomocí příkazu **runivt** .
3. Musí být nastavena cesta k rámcovým souborům .NET `wsdl.exe` a `csc.exe` . Kopie `wsdl.exe` a `csc.exe` identifikované pomocí proměnné PATH musí být na stejné úrovni rámce .NET . Pokud máte nainstalováno více rámců produktu .NET nebo používáte produkt Visual Studio, zkontrolujte pečlivě proměnnou PATH .
4. Chcete-li znovu nasadit službu:
   a. Odstraňte podadresář `./generated` a všechny jeho podadresáře.
   b. Odeberte požadavky z cílové fronty a odstraňte ji.
   c. Pokračujte podle pokynů z kroku “2” na stránce 1306.

Informace o této úloze

Tyto pokyny slouží k první implementaci služby produktu .NET . Chcete-li znovu spustit službu .NET , znovu spusťte modul listener SOAP .NET , krok “9” na stránce 1307.

Chcete-li implementovat nové služby rámce .NET Framework 1 nebo .NET Framework 2 do transportu produktu IBM MQ pro protokol SOAP, postupujte podle následujících pokynů:

Postup

1. Vytvořte adresář *deployDir* , který bude obsahovat soubory implementace.
   Obslužný program implementace vyžaduje, aby byla každá služba implementována ze samostatného adresáře.
2. Otevřete příkazové okno v produktu *deployDir* a spusťte příkaz **amqwdeployWMQService**.

```
C:\IBM\ID\QuoteClient>
```

3. Spuštěním příkazu **amqwsetcp** nastavte cestu ke třídě.
   Cesta ke třídě je potřebná pouze pro klienty Axis.
4. Zkopírujte službu .NET , *className*`.asmx`, do *deployDir*
5. Sestavte implementaci služby do knihovny (`.dll`).
   Vložená implementace služby je v produktu *className*`.asmx`. Implementace služby za provozu by mohla být *className*`.asmx.cs`.

Obrázek 192 na stránce 1307 uvádí příklad příkazu k sestavení služby .NET Framework V2 jako knihovny.

```
c:\WINDOWS\Microsoft.NET\Framework\v3.5\Csc.exe /noconfig /nowarn:1701,1702
/errorreport:prompt /warn:4 /define:TRACE
/reference:c:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\System.configuration.dll
/reference:c:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\System.Data.dll
/reference:c:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\System.dll
/reference:c:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\System.Drawing.dll
/reference:c:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\System.Web.dll
/reference:c:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\System.Web.Services.dll
/reference:c:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\System.Xml.dll
/debug:pdbonly /filealign:512 /optimize+
/out:obj\Quote.dll /target:library Properties\AssemblyInfo.cs Quote.asmx.cs
```

Obrázek 192. Příkaz sestavení pro službu .NET Framework V2

6. Zkopírujte *className*.dll do *deployDir**bin*.
7. Nastavte prostředky produktu IBM MQ a vytvořte modul listener vyžadovaný pro službu:

```
amqwdeployWMQService -f className.asmx -c genAsmxWMQBits
-v -u  "jms:/queue?destination=queueName
&initialContextFactory=com.ibm.mq.jms.Nojndi
&connectionFactory=(connectQueueManager(QmgrName)binding(auto))
&targetService=className.asmx"
```

8. Pokud potřebujete generovat kód WSDL pro službu, generovat stuby klienta nebo servery proxy klienta, spusťte **amqwdeployWMQService** s jedním z následujících parametrů:
   - genAsmxWsdl
   - genAxisWsdl
   - genProxiesToDotNet
   - genProxiestoOsa

   Poznámka: Před generováním serverů proxy je třeba vygenerovat kód WSDL.
9. Spusťte vygenerovaný modul listener produktu .NET .

```
.\generated\server\startWMQNListener.cmd
```

Související úlohy

Implementace služby na Axis 1.4 k použití pro přenos WebSphere pro SOAP pomocí produktu amqwdeployWMQService
Implementujte službu Axis 1.4 pro transport SOAP pro produkt IBM MQ vytvořením adresáře implementace, spuštěním příkazu **amqwdeployWMQService** a spuštěním modulu listener Axis 1.4 .

Implementace služby pro server CICS Transaction Server pro použití přenosu WebSphere pro SOAP
Přenos produktu IBM MQ pro SOAP je integrován do podpory webových služeb produktu CICS Transaction Server 4.1 .

Implementace služby do produktu WebSphere Application Server pro použití přenosu WebSphere pro protokol SOAP
Přenos produktu IBM MQ pro SOAP je integrován do sběrnice pro integraci služeb v produktu WebSphere Application Server.

Implementace služby pro koncový bod služby produktu WebSphere ESB a Process Server pro použití přenosu produktu WebSphere pro protokol SOAP
Přenos produktu IBM MQ pro SOAP není přímo podporován produkty WebSphere ESB a Process Server. Musíte nakonfigurovat vlastní export.

Implementace služby pro server CICS Transaction Server pro použití přenosu WebSphere pro SOAP

Přenos produktu IBM MQ pro SOAP je integrován do podpory webových služeb produktu CICS Transaction Server 4.1 .

Než začnete

Použijte stejné nástroje, které se mají vyvinout pro klienta nebo službu pro IBM MQ, jak byste se měli vyvíjet pro HTTP. CICS má nástroje odpovídající **Java2wsdl** a **wsdl2Java**:

- **DFHWS2LS** přebírá popis webové služby jako výchozí bod. Používá popisy zpráv a datové typy použité v těchto zprávách k vytvoření datových struktur jazykových dat na vysoké úrovni. Můžete je použít ve strukturách v aplikačních programech napsaných v různých jazycích.
- Produkt **DFHLS2WS** jako výchozí bod používá datovou strukturu jazyka vyšší úrovně. Používá strukturu pro konstrukci popisu webových služeb, který obsahuje popisy zpráv. Vytvoří také schémata pro zprávy ze struktury dat jazyka.

Chcete-li vytvořit webovou službu, postupujte podle pokynů Vytvoření webové služby v dokumentaci produktu CICS .

Informace o této úloze

Postupujte podle pokynů v části Konfigurace produktu CICS pro použití přenosu IBM MQ v dokumentaci produktu CICS . Pomocí pokynů můžete implementovat webovou službu do transportu produktu IBM MQ pro protokol SOAP.

Související úlohy

Implementace služby na Axis 1.4 k použití pro přenos WebSphere pro SOAP pomocí produktu amqwdeployWMQService
Implementujte službu Axis 1.4 pro transport SOAP pro produkt IBM MQ vytvořením adresáře implementace, spuštěním příkazu **amqwdeployWMQService** a spuštěním modulu listener Axis 1.4 .

Implementace služby pro službu .NET Framework 1 nebo 2 pro použití přenosu produktu IBM MQ pro protokol SOAP
Implementujte službu .NET Framework 1 nebo 2 do transportu produktu IBM MQ pro protokol SOAP. Vytvořte adresář implementace, spusťte příkaz **amqwdeployWMQService** a spusťte modul listener produktu .NET .

Implementace služby do produktu WebSphere Application Server pro použití přenosu WebSphere pro protokol SOAP
Přenos produktu IBM MQ pro SOAP je integrován do sběrnice pro integraci služeb v produktu WebSphere Application Server.

Implementace služby pro koncový bod služby produktu WebSphere ESB a Process Server pro použití přenosu produktu WebSphere pro protokol SOAP
Přenos produktu IBM MQ pro SOAP není přímo podporován produkty WebSphere ESB a Process Server. Musíte nakonfigurovat vlastní export.

Implementace služby do produktu WebSphere Application Server pro použití přenosu WebSphere pro protokol SOAP

Přenos produktu IBM MQ pro SOAP je integrován do sběrnice pro integraci služeb v produktu WebSphere Application Server.

Než začnete

Pomocí produktu Rational Application Developer, WebSphere Integration Developer nebo sady nástrojů webových služeb vytvořte webovou službu.

Informace o této úloze

Pomocí následujících pokynů implementujte službu pomocí transportu produktu IBM MQ pro protokol SOAP jako přenos SOAP v produktu WebSphere Application Server.

Postup

1. Nakonfigurujte produkt IBM MQ jako poskytovatele systému zpráv produktu JMS pro sběrnici pro integraci služeb v produktu WebSphere Application Server.
2. Konfigurujte prostředky produktu IBM MQ vyžadované službou.
3. Postupujte podle pokynů v dokumentaci produktu WebSphere Application Server Network Deployment v části Konfigurace prostředků produktu JMS pro synchronní protokol SOAP over JMS.

   Pro další platformy WebSphere Application Server existují odpovídající pokyny.
4. Upravte identifikátor URI služby tak, aby odpovídal transportu produktu IBM MQ pro identifikátor URI protokolu SOAP.
5. Implementujte službu do produktu WebSphere Application Server.

Jak pokračovat dále

Implementujte službu pomocí protokolu HTTP jako transport, aby se klienti mohli dotazovat na službu a přijímat WSDL v odezvě.

Související úlohy

Implementace služby na Axis 1.4 k použití pro přenos WebSphere pro SOAP pomocí produktu amqwdeployWMQService

Implementujte službu Axis 1.4 pro transport SOAP pro produkt IBM MQ vytvořením adresáře implementace, spuštěním příkazu **amqwdeployWMQService** a spuštěním modulu listener Axis 1.4 .

Implementace služby pro službu .NET Framework 1 nebo 2 pro použití přenosu produktu IBM MQ pro protokol SOAP

Implementujte službu .NET Framework 1 nebo 2 do transportu produktu IBM MQ pro protokol SOAP. Vytvořte adresář implementace, spusťte příkaz **amqwdeployWMQService** a spusťte modul listener produktu .NET .

Implementace služby pro server CICS Transaction Server pro použití přenosu WebSphere pro SOAP

Přenos produktu IBM MQ pro SOAP je integrován do podpory webových služeb produktu CICS Transaction Server 4.1 .

Implementace služby pro koncový bod služby produktu WebSphere ESB a Process Server pro použití přenosu produktu WebSphere pro protokol SOAP

Přenos produktu IBM MQ pro SOAP není přímo podporován produkty WebSphere ESB a Process Server. Musíte nakonfigurovat vlastní export.

Implementace služby pro koncový bod služby produktu WebSphere ESB a Process Server pro použití přenosu produktu WebSphere pro protokol SOAP

Přenos produktu IBM MQ pro SOAP není přímo podporován produkty WebSphere ESB a Process Server. Musíte nakonfigurovat vlastní export.

Informace o této úloze

Produkt WebSphere Integration Developer nabízí transformaci dat SOAP, kterou lze svázat s exportem produktu IBM MQ JMS a vytvořit vlastní export SOAP produktu IBM MQ JMS .

Postupujte podle pokynů pro vytvoření upraveného exportu pro příjem požadavků SOAP prostřednictvím transportu produktu IBM MQ pro protokol SOAP.

Postup

1. Přečtěte si téma "Přehled importů a exportů" a "Jak se připojit k produktu IBM MQ" v dokumentaci k produktu WebSphere Process Server for Multiplatforms.

Viz Přehled importů a exportů a Jak se připojit k produktu IBM MQ.

2. Postupujte podle úlohy "Generování vazby exportu produktu MQ JMS" v dokumentaci produktu WebSphere Integration Developer.

   Viz téma Generování vazby exportu produktu MQ JMS. Chcete-li formátovat zprávu SOAP, použijte předem zabalenou vazbu dat SOAP popsanou v části Předem seskupené transformace formátu dat produktu JMS .

Související úlohy

Implementace služby na Axis 1.4 k použití pro přenos WebSphere pro SOAP pomocí produktu amqwdeployWMQService
Implementujte službu Axis 1.4 pro transport SOAP pro produkt IBM MQ vytvořením adresáře implementace, spuštěním příkazu **amqwdeployWMQService** a spuštěním modulu listener Axis 1.4 .

Implementace služby pro službu .NET Framework 1 nebo 2 pro použití přenosu produktu IBM MQ pro protokol SOAP
Implementujte službu .NET Framework 1 nebo 2 do transportu produktu IBM MQ pro protokol SOAP. Vytvořte adresář implementace, spusťte příkaz **amqwdeployWMQService** a spusťte modul listener produktu .NET .

Implementace služby pro server CICS Transaction Server pro použití přenosu WebSphere pro SOAP
Přenos produktu IBM MQ pro SOAP je integrován do podpory webových služeb produktu CICS Transaction Server 4.1 .

Implementace služby do produktu WebSphere Application Server pro použití přenosu WebSphere pro protokol SOAP
Přenos produktu IBM MQ pro SOAP je integrován do sběrnice pro integraci služeb v produktu WebSphere Application Server.

Implementace klientů webových služeb pro použití přenosu produktu IBM MQ pro protokol SOAP

Nasaďte klienta webové služby na jeden z mnoha různých klientských prostředí a připojte se ke službě pomocí přenosu IBM MQ pro SOAP.

Než začnete

Vytvořte webovou službu a implementujte ji pro použití přenosu produktu IBM MQ pro protokol SOAP.

Informace o této úloze

Klienta webové služby můžete implementovat tak, aby se spouštěl s přenosem IBM MQ pro protokol SOAP v řadě různých klientských prostředí. Klienta produktu Java můžete implementovat na Axis 1.4 pouze s použitím softwaru instalovaného s produktem IBM MQ. V případě ostatních klientských prostředí je třeba instalovat další software.

Pro spuštění přenosu produktu WebSphere pro protokol SOAP v prostředí klienta, pro které existují pokyny k implementaci, nejste omezeni. Použijte pokyny k implementaci klienta do jednoho z podporovaných prostředí.

Poznámka: Některá integrovaná prostředí nabízejí protokol SOAP prostřednictvím produktu JMS s doporučenou vazbou JMS protokolu SOAP W3C a také přenosem IBM MQ pro vazbu SOAP. Vydání produktu IBM MQ, včetně 7.0.1.2, podporuje pouze přenos IBM MQ pro vazbu SOAP. Počínaje verzí 7.0.1.3 můžete implementovat klienty Axis2 pomocí identifikátoru URI, který odpovídá doporučenému doporučení W3C pro protokol SOAP prostřednictvím produktu JMS. Viz výukový program Vývoj aplikací webových služeb SOAP/JMS JAX-WS s WebSphere Application Server V7 a Rational Application Developer 7.5.

Implementace klienta webové služby do Axis 1.4 pro použití přenosu IBM MQ pro SOAP

Připravte adresář implementace a deskriptor implementace pro klienta. Zadejte proxy klienta a třídu klienta a nastavte proměnnou prostředí CLASSPATH. Nakonfigurujte fronty a kanály produktu IBM MQ , spusťte danou službu a otestujte klienta.

Než začnete

Tip: Implementujte službu do protokolu HTTP, vyvinete a otestujte klienta pro protokol HTTP a poté upravíte klienta pro přenos IBM MQ pro protokol SOAP:

1. Přidejte volání Register.extension() do klienta.
2. Změňte adresu statické webové služby ve třídě lokátoru proxy klienta tak, aby používala identifikátor URI pro přenos IBM MQ pro protokol SOAP.

Informace o této úloze

Implementace klienta Axis 1.4 pro použití přenosu IBM MQ pro SOAP vyžaduje jeden další krok implementace ve srovnání s klientem HTTP. Chcete-li mapovat transport `jms:` na třídu odesílatele `com.ibm.mq.soap.transport.jms.WMQSender`, musíte vytvořit deskriptor implementace klienta `client-config.wsdd`.

Pokud používáte příkaz **amqwdeployWMQService** ke generování klientských serverů proxy, můžete implementovat klienta pomocí adresářů, které příkaz vygeneruje.

Postup

1. Vytvořte adresář *deployDir* , kde budou uloženy soubory implementace klienta.
2. Otevřete příkazové okno v systémech Windows nebo příkazový shell pomocí systému X Window System na systémech UNIX and Linux , v adresáři *deployDir*.
3. Spuštěním příkazu **amqwsetcp.cmd** nastavte proměnnou prostředí CLASSPATH .
4. Spuštěním příkazu **amqwclientconfig.cmd** vytvořte deskriptor implementace klienta Axis 1.4 , `client-config.wsdd` v adresáři *deployDir*.
5. Ujistěte se, že třídy v balíku klienta, třídy proxy klienta a knihovny, které klient používá, jsou uvedeny v proměnné CLASSPATH.

   Produkt **amqwdeployWMQService** umístí proxy klienta produktu .NET do produktu `./generated/server/soap/client/remote/dotnetService` a proxy Axis 1.4 do produktu `./generated/server/soap/client/remote/`*`client package`*.

Příklad

Příklad ukazuje konfiguraci a výstup, Obrázek 195 na stránce 1312, z klienta Axis 1.4 Java . Klient Obrázek 194 na stránce 1312volá webovou službu, která odráží svůj vstupní parametr. Definice služby, Obrázek 193 na stránce 1311, ukazuje identifikátor URI převzaté ze souboru WSDL služby.

```
<wsdl:service name="QuoteSOAPImplService">
    wsdl:port binding="intf:org.example.www.QuoteSOAPImplBindingSoap"
                      name="org.example.www.QuoteSOAPImpl_Wmq">
       <wsdlsoap:address location="jms:/queue?destination=REQUESTAXIS
           &connectionFactory=(connectQueueManager(QM1)binding(server))
           &initialContextFactory=com.ibm.mq.jms.Nojndi
           &targetService=org.example.www.QuoteSOAPImpl.java
           &replyDestination=SYSTEM.SOAP.RESPONSE.QUEUE" />
    </wsdl:port>
</wsdl:service>
```

Obrázek 193. definice služby

```
package org.example.www;
import com.ibm.mq.soap.Register;
public class QuoteClient {
   public static void main(String[] args) {
      try {
         Register.extension();
         QuoteSOAPImplServiceLocator locator = new QuoteSOAPImplServiceLocator();
         System.out.println("Response = "
                  + locator.getOrgExampleWwwQuoteSOAPImpl_Wmq().getQuote("IBM"));
      } catch (Exception e) {
         System.out.println("Exception = " + e.getMessage());
      }
   }
}
```

Obrázek 194. Klient Axis 1.4 Java

```
C:\IBM\ID\Test>dir /s /b
C:\IBM\ID\Test\client-config.wsdd
C:\IBM\ID\Test\org
C:\IBM\ID\Test\org\example
C:\IBM\ID\Test\org\example\www
C:\IBM\ID\Test\org\example\www\GetQuoteFaultMsg.class
C:\IBM\ID\Test\org\example\www\OrgExampleWwwQuoteSOAPImplBindingSoapStub.class
C:\IBM\ID\Test\org\example\www\QuoteClient.class
C:\IBM\ID\Test\org\example\www\QuoteSOAPImpl.class
C:\IBM\ID\Test\org\example\www\QuoteSOAPImplService.class
C:\IBM\ID\Test\org\example\www\QuoteSOAPImplServiceLocator.class

C:\IBM\ID\Test>amqwsetcp
C:\IBM\ID\Test>java org.example.www.QuoteClient.class
Response = IBM
```

Obrázek 195. Konfigurace a výstup klienta

Jak pokračovat dále

1. Implementujete-li klienta jako klienta produktu IBM MQ , nakonfigurujte kanál připojení klienta a serveru.
2. Implementujete-li klienta do jiného správce front do služby, je třeba zpřístupnit cílovou frontu klientovi. Konfigurujte cílovou frontu ve správci front služeb jako frontu klastru nebo ve správci front klienta jako definici vzdálené fronty.

Související úlohy

Implementace klienta webové služby do prostředí Axis2 pro použití transportu produktu IBM MQ pro protokol SOAP

Připravte adresář implementace a konfigurační soubor Axis2 pro klienta. Zadejte proxy klienta a třídu klienta a nastavte proměnnou CLASSPATH. Nakonfigurujte fronty a kanály produktu IBM MQ , spusťte danou službu a otestujte klienta.

Implementace na klienta Axis2 pomocí protokolu W3C SOAP prostřednictvím produktu JMS

Webová služba, která je svázána s doporučujícím doporučením W3C pro protokol SOAP over JMS , musí být spuštěna v kontejneru EJB aplikačního serveru Java EE. Tato úloha je krokem 4 připojení klienta webové služby Axis2 a webové služby implementované do produktu WebSphere Application Server pomocí protokolu W3C protokolu SOAP prostřednictvím protokolu JMS . Upravte adresu URL v klientovi Axis2 vyvinutém pro transport IBM MQ pro protokol SOAP tak, aby bylo možné použít doporučení kandidáta W3C pro protokol SOAP prostřednictvím produktu JMS.

Implementace klienta webové služby do produktu .NET Framework 1 a 2 pro použití přenosu produktu IBM MQ pro protokol SOAP

Připravte adresář implementace a deskriptor implementace pro klienta. Zadejte proxy klienta a třídu klienta. Nakonfigurujte fronty a kanály produktu IBM MQ , spusťte danou službu a otestujte klienta.

Implementace klienta webové služby do prostředí Axis2 pro použití transportu produktu IBM MQ pro protokol SOAP

Připravte adresář implementace a konfigurační soubor Axis2 pro klienta. Zadejte proxy klienta a třídu klienta a nastavte proměnnou CLASSPATH. Nakonfigurujte fronty a kanály produktu IBM MQ , spusťte danou službu a otestujte klienta.

Než začnete

Tip: Implementujte službu do protokolu HTTP. Vyviňte a otestujte klienta pro protokol HTTP a poté upravte adresu URL tak, aby odkazovaly na službu pomocí transportu produktu IBM MQ pro protokol SOAP.

Tato úloha ukazuje, jak implementovat nespravovaný klient Axis2 do produktu Java Standard Edition. Je možné, že budete chtít implementovat klienta Axis2 do webového kontejneru. V produktu “Vyvíjení klienta JAX-WS pro produkt WebSphere transport pro protokol SOAP s použitím prostředí Eclipse” na stránce 1285jste vyvinuli klienta ve webovém kontejneru a implementovali jste jej na produkt WebSphere Application Server Community Edition. Jako součást konfigurace serveru jste povolili fasetu Axis2 a zahrnuli jste fasetu v konfiguraci webového kontejneru. Chcete-li nakonfigurovat webové kontejnery na jiných aplikačních serverech, přečtěte si dokumentaci Axis2 , https://ws.apache.org/axis2/1_4_1/installationguide.html#servlet_containernebo dokumentaci dodanou s webovým serverem.

Poznámka: Axis2 používá termín, kontejner servletu. Kontejner servletů je stejný jako webový kontejner.

Informace o této úloze

Implementace klienta Axis2 pro použití transportu produktu IBM MQ pro protokol SOAP je jako implementace klienta Axis2 pro použití protokolu HTTP. Další kroky jsou vyžadovány k poskytnutí cesty ke třídě pro soubory JAR produktu IBM MQ a k úpravě konfiguračního souboru Axis2 . Konfigurační soubor Axis2 vyžaduje další položku pro produkt JMS. Položka odkazuje na přenos IBM MQ pro soubor JAR SOAP, který implementuje JMS `transportSender`.

Axis2 poskytuje skript, `axis2.bat` nebo `axis2.sh`, který zjednodušuje implementaci klienta; viz příklady v Obrázek 199 na stránce 1315 a Obrázek 200 na stránce 1315.

Poznámka:

1. `axis2.bat` má chybu, která musí být opravena. Řetězec `-Djava.ext.dirs="%AXIS2_HOME%\lib\"` musí být změněn na `-Djava.ext.dirs="%AXIS2_HOME%\lib\\"`.
2. V produktu `axis2.bat` a `axis2.sh`se produkt `-Djava.ext.dirs` používá jako rychlý způsob, jak odkazovat na všechny soubory JAR Axis2 místo jejich přidání do cesty ke třídě. Tento přístup je bohužel chybný a funguje pouze s některými prostředími JRE. Nepracuje s JRE produktu IBM .

   Parametr prostředí JVM `-Djava.ext.dirs="%AXIS2_HOME%\lib\\"`zpřístupňuje soubory JAR Axis pro prostředí JVM. Prostředí JVM se pokouší o vytvoření instance některých souborů JAR Axis a vede k chybě, jejíž podrobnosti závisejí na prostředí JVM. Obvykle se v trasování zásobníku může zobrazit jeden z následujících řádků:

   ```
   org.apache.axiom.om.util.UUIDGenerator.getInitialUUID(UUIDGenerator.java:76)
   ```

   , nebo `org.apache.axis2.deployment.DeploymentException: java.security.NoSuchAlgorithmException: MD5 MessageDigest not available`

   Správným způsobem spuštění nespravovaného klienta Axis2 je přidání souborů JAR Axis2 do cesty ke třídám. Cesta ke třídám je k dispozici pouze pro aplikaci klienta a nikoli pro prostředí JVM.

Procedura popisuje obecné kroky ke spuštění nespravovaného klienta Axis2 bez použití skriptu axis2 . Příklady v Obrázek 197 na stránce 1315 a Obrázek 198 na stránce 1315 jsou skripty pro Windows a Linux.

Postup

1. Stáhněte si Axis2 1.4.1 z webu https://ws.apache.org/axis2/download/1_4_1/download.cgi a rozbalte ji do složky `Axis2-1.4.1`.
2. Aktualizujte produkt `axis2.xml` v produktu `Axis2-1.4.1\conf`.
   a) Aktualizujte produkt `axis2.xml` v produktu `Axis2-1.4.1\conf`. Přidejte přenos IBM MQ pro SOAP jako `transportSender`:

   ```
   <transportSender name="jms"
   class="com.ibm.mq.axis2.transport.jms.WMQJMSTransportSender"/>
   ```

   b) V případě potřeby změňte velikost fondu připojení z výchozího nastavení 10.

   ```
   <transportSender name="jms"
   class="com.ibm.mq.axis2.transport.jms.WMQJMSTransportSender">
   <parameter name="ResourcePoolCapacity">20</parameter>
   </transportSender>
   ```

   Položka `ResourcePoolCapacity` definuje, kolik položek koncového bodu služby je uchováno v mezipaměti. Hodnota musí být alespoň 1. Pokud počet položek koncového bodu služby překročí velikost mezipaměti, položky se odstraní, aby se místo nových záznamů neslo o místo. Velikost záznamu koncového bodu se mění. Nastavte číslo, které je dostatečně velké, aby se zabránilo hluchému stránkování mezipaměti.

   Viz krok 3 v příručce "Vyvíjení klienta JAX-WS pro produkt WebSphere transport pro protokol SOAP s použitím prostředí Eclipse" na stránce 1285.
3. Vytvořte adresář *deployDir*. Pod tímto adresářem zkopírujte strukturu složek obsahující proxy klienta a klienta. *deployDir* je ekvivalentem složky *project*`\bin` v projektu Eclipse Java .
4. Otevřete příkazové okno na systému Windowsnebo shell příkazů pomocí systému X Window System na systémech UNIX and Linux v adresáři *deployDir*.
5. Aktualizujte cestu ke třídě, aby zahrnovala aktuální adresář, Axis2 soubory JAR, `com.ibm.mqjms.jar` a `com.ibm.mq.axis2.jar`.

   `com.ibm.mqjms.jar` odkazuje na všechny ostatní soubory JAR IBM MQ , které jsou povinné.
6. Chcete-li spustit klientský program, použijte příkaz **Java** .

Příklady

Čtyři příklady spuštění klienta Axis2 jsou uvedeny v seznamu Obrázek 198 na stránce 1315 na hodnotu Obrázek 200 na stránce 1315. Obrázek 196 na stránce 1314 zobrazuje výstup od spuštění asynchronního klienta uvedeného v seznamu Obrázek 179 na stránce 1290.

```
cd C:\IBM\ID\Workspaces\Axis2docs\StockQuoteAxis2PojoClient\bin>
runpojo soap/client/SQA2AsyncClient

HTTP Sync: 55.25
Callback constructor
Waiting for HTTP callback
Result in Callback 55.25
HTTP poll: 55.25
JMS: Sync: 55.25
Callback constructor
Waiting for JMS callback
Result in Callback 55.25
JMS poll: 55.25
Press any key to continue . . .
```

Obrázek 196. Výstup ze spuštění SQA2AsyncClient

```
@echo off
set AXIS2_HOME=C:\OpenSource\axis2-1.4.1
set JAVA_HOME=C:\IBM\Java50
set WMQ_HOME=C:\IBM\MQ\java\lib

setlocal EnableDelayedExpansion
set CLASSPATH=
set AXIS2_CLASS_PATH=
FOR %%c in ("%AXIS2_HOME%\lib\*.jar") DO set AXIS2_CLASS_PATH=!AXIS2_CLASS_PATH!;%%c

"%JAVA_HOME%\bin\java" -Daxis2.repo="%AXIS2_HOME%\repository"
-Daxis2.xml="%AXIS2_HOME%\conf\axis2.xml" -cp
".;%WMQ_HOME%\com.ibm.mqjms.jar;%WMQ_HOME%\com.ibm.mq.axis2.jar;%AXIS2_CLASS_PATH%" %1

pause
```

Obrázek 197. runpojo.bat: Windowspomocí cesty ke třídě

```
export AXIS2_HOME=/home/OpenSource/axis2-1.4.1
export JAVA_HOME=/usr/lib/j2sdk1.5-ibm
# update classpath
AXIS2_CLASSPATH=""
for f in "$AXIS2_HOME"/lib/*.jar
do
  AXIS2_CLASSPATH="$AXIS2_CLASSPATH":$f
done
AXIS2_CLASSPATH="$AXIS2_HOME":"$JAVA_HOME/lib/tools.jar":"$AXIS2_CLASSPATH":"$CLASSPATH"
java -cp /home/alex/dev/sandbox/Soap/axis2:/opt/mqm/java/lib/com.ibm.mqjms.jar:
/opt/mqm/java/lib/com.ibm.mq.axis2.jar:$AXIS2_CLASSPATH
-Daxis2.xml=/home/alex/dev/sandbox/axis2-1.4.1/conf/axis2.xml %1
```

Obrázek 198. runpojo.sh: Linuxpomocí cesty ke třídě.

```
@echo off
set AXIS2_HOME=C:\OpenSource\axis2-1.4.1
set JAVA_HOME=C:\IBM\Java50
set WMQ_HOME=C:\IBM\MQ\java\lib

%AXIS2_HOME%\bin\axis2 -cp .;%WMQ_HOME%\com.ibm.mqjms.jar;%WMQ_HOME%\com.ibm.mq.axis2.jar; %1
pause
```

Obrázek 199. runaxis2.bat: Windows, použití axis2.bat

Poznámka

```
export AXIS2_HOME=/home/OpenSource/axis2-1.4.1
export JAVA_HOME=/usr/lib/j2sdk1.5-ibm

%AXIS2_HOME%\bin\axis2 -cp .;%WMQ_HOME%\com.ibm.mqjms.jar;%WMQ_HOME%\com.ibm.mq.axis2.jar; %1
```

Obrázek 200. runaxis2.sh: Linux, použití axis2.sh

Poznámka

Související úlohy

Implementace klienta webové služby do Axis 1.4 pro použití přenosu IBM MQ pro SOAP

Připravte adresář implementace a deskriptor implementace pro klienta. Zadejte proxy klienta a třídu klienta a nastavte proměnnou prostředí CLASSPATH. Nakonfigurujte fronty a kanály produktu IBM MQ , spusťte danou službu a otestujte klienta.

Implementace na klienta Axis2 pomocí protokolu W3C SOAP prostřednictvím produktu JMS

Webová služba, která je svázána s doporučujícím doporučením W3C pro protokol SOAP over JMS , musí být spuštěna v kontejneru EJB aplikačního serveru Java EE. Tato úloha je krokem 4 připojení klienta webové služby Axis2 a webové služby implementované do produktu WebSphere Application Server pomocí protokolu W3C protokolu SOAP prostřednictvím protokolu JMS . Upravte adresu URL v klientovi Axis2 vyvinutém pro transport IBM MQ pro protokol SOAP tak, aby bylo možné použít doporučení kandidáta W3C pro protokol SOAP prostřednictvím produktu JMS.

Implementace klienta webové služby do produktu .NET Framework 1 a 2 pro použití přenosu produktu IBM MQ pro protokol SOAP
Připravte adresář implementace a deskriptor implementace pro klienta. Zadejte proxy klienta a třídu klienta. Nakonfigurujte fronty a kanály produktu IBM MQ , spusťte danou službu a otestujte klienta.

Implementace na klienta Axis2 pomocí protokolu W3C SOAP prostřednictvím produktu JMS

Webová služba, která je svázána s doporučujícím doporučením W3C pro protokol SOAP over JMS , musí být spuštěna v kontejneru EJB aplikačního serveru Java EE. Tato úloha je krokem 4 připojení klienta webové služby Axis2 a webové služby implementované do produktu WebSphere Application Server pomocí protokolu W3C protokolu SOAP prostřednictvím protokolu JMS . Upravte adresu URL v klientovi Axis2 vyvinutém pro transport IBM MQ pro protokol SOAP tak, aby bylo možné použít doporučení kandidáta W3C pro protokol SOAP prostřednictvím produktu JMS.

Než začnete

Nejprve musíte dokončit úlohu, “Vyvíjení klienta JAX-WS pro produkt WebSphere transport pro protokol SOAP s použitím prostředí Eclipse” na stránce 1285 pro volání **`SimpleJavaListener`** pomocí klienta Axis2 a přenosu IBM MQ pro protokol SOAP.

Musíte také vytvořit webovou službu a nakonfigurovat produkt IBM MQ a aplikační server WebSphere .

Viz Použití produktů IBM MQ a WebSphere Application Server společně.

Musíte také dokončit následující úlohu, “Vývoj webové služby JAX-WS pro W3C SOAP přes JMS” na stránce 1272.

V rámci úlohy je klient spuštěn v systému Eclipse Galileo. Klienta můžete spustit z příkazového řádku tak, že upravíte soubor `Axis2.bat` dodávaný s Axis2.

Informace o této úloze

Jediná změna, kterou musíte provést na stávajícím statickém klientovi Axis2 `StockQuoteAxis` pro volání služby Axis služby StockQuotehostované produktem WebSphere Application Server , je změna adresy URL předané klientovi. Vzhledem k tomu, že se WSDL nezměnil, můžete použít stejné třídy proxy v balíku `soap.server` .

K definování adresy URL, která má být předána klientovi, máte dva přístupy. Můžete použít stejnou adresu URL jako v generovaném souboru `StockQuoteAxis.wsdl`. Musíte přidat parametry `jndiInitialContextFactory` a `jndiURL` pro přístup k adresáři JNDI WebSphere Application Server . Jiným přístupem je změna adresy URL a poskytnutí přímého přístupu klienta k frontám `REQUESTAXIS` a `REPLYAXIS` na serveru `QM1`bez použití vyhledávání JNDI.

Parametry připojení, které definujete v adrese URL předané klientovi Axis2 , se používají pro připojení ke správci front produktu IBM MQ a k frontám, které jsou vyžadovány k odesílání a přijímání zpráv SOAP. Parametry připojení předané klientovi Axis2 nemusí být nutně používány službou. K odpojení klienta a služby od použití stejného správce front nebo stejného serveru názvů můžete použít funkce distribuované fronty v produktu IBM MQ .

Postup

1. Uložte adresu URL z generované `StockQuoteAxis.wsdl` a zavřete produkt Rational Application Developer, abyste jej uložili do paměti.

Pokud jste nezměnili konfiguraci serveru, zavřením produktu Rational Application Developer se aplikační server zastaví. V takovém případě spusťte server s příkazem:

```
startserver server1
```

2. Otevřete projekt Eclipse Galileo v pracovním prostoru s klientským projektem Axis2 .
3. Otevřete produkt `SQA2StaticClient.java`.

   Viz SQA2StaticClient.java.
4. Volejte službu pomocí varianty `queue` s použitím varianty identifikátoru URI.
   a) Upravte adresu URL.

   Nový URI je:

   ```
   jms:queue:REQUESTAXIS
             ?replyToName=REPLYAXIS
             &connectionFactory=connectQueueManager(QM1)Bind(Server)
             &targetService=StockQuoteAxis;
   ```

   Porovnejte je s adresou URL z produktu `StockQuoteAxis.wsdl`:

   ```
   jms:jndi:requestaxis
            ?jndiConnectionFactoryName=qm1
            &targetService=StockQuoteAxis
   ```

   Obrázek 201. Adresa URL od `StockQuoteAxis.wsdl`

   - REQUESTAXIS je nyní velká písmena, protože jde o název fronty a nikoli na název rozhraní JNDI.
   - Připojení k produktu `QM1` je jednoduché.
   - Identifikátor URI neobsahuje název odpovědi na místo určení. Klient musí definovat frontu, na které očekává odpovědi.

   b) Spusťte produkt `SQA2StaticClient.java` s použitím stejného příkazu **Spustit jako ...** konfigurace, jak jste provedli v úloze “Vyvíjení klienta JAX-WS pro produkt WebSphere transport pro protokol SOAP s použitím prostředí Eclipse” na stránce 1285.
5. Volejte službu pomocí `jndi` varianty identifikátoru URI a použijte WebSphere Application Server jako server názvů.
   a) Použijte adresu URL z produktu `StockQuoteAxis.wsdl`, Obrázek 201 na stránce 1317, čímž poskytnete chybějící parametry pro použití služby názvů v produktu WebSphere Application Server.

   Chybějící parametry a hodnoty, které musíte zadat, jsou:

Tabulka 192. Další parametry JNDI

Parametr	Hodnota použitá v tomto příkladu	Popis
`&jndiURL`	`iiop://localhost:2810`, nebo `corbaname:iiop:localhost:2810`	Identifikátor URI poskytovatele názvů. Pro WebSphere Application Server je výchozí hodnota na 2809. Je také znám jako číslo portu konektoru RMI a port samozavedení. Hodnota je uvedena v seznamu `SystemOut.log` `00000000 NameServerImp A NMSV0018I: Name server available on bootstrap port 2810`

Tabulka 192. Další parametry JNDI (pokračování)		
Parametr	**Hodnota použitá v tomto příkladu**	**Popis**
`&jndiInitialContextFactory`	`com.ibm.websphere.naming.WsnInitialContextFactory`	Název počáteční kontextové továrny používané produktem WebSphere Application Server.
`&replyToName`	`replyaxis`	Název rozhraní JNDI fronty `REPLYAXIS` .

```
jms:jndi:requestaxis?
         &jndiURL=iiop//localhost:2810
         &jndiConnectionFactoryName=qm1
         &jndiInitialContextFactory=com.ibm.websphere.naming.WsnInitialContextFactory
         &targetService=StockQuoteAxis
         &replyToName=replyaxis;
```

b) Přidejte soubory JAR požadované vyhledáváním v rozhraní JNDI.

V této konfiguraci byly do cesty sestavení přidány následující soubory JAR, aby bylo možné spustit úlohu pomocí varianty `jndi` adresy URL JMS :

- `com.ibm.jaxws.thinclient_7.0.0.jar` z *`Rational installation directory`*`\SDP\runtimes\base_v7\runtimes`.
- `com.ibm.ws.runtime.jar` z *`Rational installation directory`*`\SDP\runtimes\base_v7\plugins`

Pro jiného poskytovatele rozhraní JNDI je třeba použít různé soubory JAR.

Další soubory JAR v cestě sestavení jsou:

i) Všechny soubory JAR v produktu *`WebSphere MQ Installation directory`*`\java\lib`.

ii) Všechny soubory JAR v produktu *`Axis2-1.5.1`*`\lib`.

iii) Java 6.0 JRE.

c) Spusťte produkt `SQA2StaticClient.java` s použitím stejného příkazu **Spustit jako ...** konfigurace, jak jste provedli v úloze “Vyvíjení klienta JAX-WS pro produkt WebSphere transport pro protokol SOAP s použitím prostředí Eclipse” na stránce 1285.

Výsledky

V obou případech se odpověď ze služby zobrazí v pohledu konzoly klienta.

Související úlohy

Implementace klienta webové služby do Axis 1.4 pro použití přenosu IBM MQ pro SOAP
Připravte adresář implementace a deskriptor implementace pro klienta. Zadejte proxy klienta a třídu klienta a nastavte proměnnou prostředí CLASSPATH. Nakonfigurujte fronty a kanály produktu IBM MQ , spusťte danou službu a otestujte klienta.

Implementace klienta webové služby do prostředí Axis2 pro použití transportu produktu IBM MQ pro protokol SOAP
Připravte adresář implementace a konfigurační soubor Axis2 pro klienta. Zadejte proxy klienta a třídu klienta a nastavte proměnnou CLASSPATH. Nakonfigurujte fronty a kanály produktu IBM MQ , spusťte danou službu a otestujte klienta.

Implementace klienta webové služby do produktu .NET Framework 1 a 2 pro použití přenosu produktu IBM MQ pro protokol SOAP
Připravte adresář implementace a deskriptor implementace pro klienta. Zadejte proxy klienta a třídu klienta. Nakonfigurujte fronty a kanály produktu IBM MQ , spusťte danou službu a otestujte klienta.

Implementace klienta webové služby do produktu .NET Framework 1 a 2 pro použití přenosu produktu IBM MQ pro protokol SOAP

Připravte adresář implementace a deskriptor implementace pro klienta. Zadejte proxy klienta a třídu klienta. Nakonfigurujte fronty a kanály produktu IBM MQ , spusťte danou službu a otestujte klienta.

Než začnete

Tip: Vytvořte a otestujte službu a klienta pomocí produktu Visual Studio. Poté upravte klienta pro transport IBM MQ pro protokol SOAP.

1. Implementujete-li službu pomocí .NET Framework 1 nebo 2, sestavte službu jako knihovnu (`.dll`). Implementujte pomocí přenosu produktu IBM MQ pro protokol SOAP.
2. Přidejte volání Register.Extension() do klienta.
3. Přidejte odkaz na soubor amqsoap.dll, který se nachází v produktu *MQ_Install*`\bin`.
4. Změňte statickou vlastnost `Url` v konstruktoru třídy proxy klienta na identifikátor URI produktu `jms:/` , pro transport IBM MQ pro protokol SOAP.

Informace o této úloze

Implementace klienta webové služby pro produkt .NET Framework 1 nebo 2 k použití transportu produktu IBM MQ pro protokol SOAP vyžaduje další krok implementace. Musíte registrovat `amqsoap.dll` s rámcem .NET Framework. Produkt `amqsoap.dll` je automaticky registrován jako součást instalace transportu produktu IBM MQ pro protokol SOAP, ale možná jej budete muset registrovat znovu.

Pokud používáte příkaz **amqwdeployWMQService** ke generování klientských serverů proxy, můžete implementovat klienta pomocí adresářů, které příkaz vygeneruje.

Postup

1. Vytvořte adresář *deployDir* , kde budou uloženy soubory implementace klienta.
2. Otevřete příkazové okno v adresáři *deployDir*.
3. Spuštěním příkazu **amqwsetcp** nastavte proměnnou prostředí `CLASSPATH` , pokud má být služba spuštěna na ose Axis 1.4.
4. V případě potřeby spusťte **amqwRegisterDotNet** pro registraci `amqsoap.dll` s rámcem .NET Framework.

Příklad

Příklad ukazuje konfiguraci a výstup Obrázek 204 na stránce 1320z klienta .NET Framework V2 . Klient Obrázek 203 na stránce 1320volá webovou službu, která odráží svůj vstupní parametr. Statická definice URL, Obrázek 202 na stránce 1319, ukazuje konstruktor pro proxy klienta.

```
public Quote() {
    this.Url = "jms:/queue?destination=REQUESTDOTNET
               &connectionFactory=(connectQueueManager(QM1)binding(server))
               &initialContextFactory=com.ibm.mq.jms.Nojndi
               &targetService=Quote.asmx
               &replyDestination=SYSTEM.SOAP.RESPONSE.QUEUE";
}
```

Obrázek 202. Statický konstruktor klienta proxy

```
using System;
namespace QuoteClientProgram {
    class QuoteMain {
        static void Main(string[] args) {
            try {
                IBM.WMQSOAP.Register.Extension();
                Quote q = new Quote();
                Console.WriteLine("Response is: " + q.getQuote("ibm"));
            } catch (Exception e) {
                Console.WriteLine("Exception is: " + e);
            }
        }
    }
}
```

Obrázek 203. Klientský program

```
C:\IBM\ID\DotNet\QuoteClientProgram\QuoteClientProgram>dir /s /b
C:\IBM\ID\DotNet\QuoteClientProgram\QuoteClientProgram\QuoteClientProgram.exe
C:\IBM\ID\DotNet\QuoteClientProgram\QuoteClientProgram>quoteclientprogram
Response is: IBM
```

Obrázek 204. Konfigurace a výstup

Jak pokračovat dále

1. Implementujete-li klienta jako IBM MQ MQI client, nakonfigurujte kanál připojení klienta a serveru.
2. Implementujete-li klienta do jiného správce front do služby, je třeba zpřístupnit cílovou frontu klientovi. Konfigurujte cílovou frontu ve správci front služeb jako frontu klastru nebo ve správci front klienta jako definici vzdálené fronty.

Související úlohy

Implementace klienta webové služby do Axis 1.4 pro použití přenosu IBM MQ pro SOAP
Připravte adresář implementace a deskriptor implementace pro klienta. Zadejte proxy klienta a třídu klienta a nastavte proměnnou prostředí CLASSPATH. Nakonfigurujte fronty a kanály produktu IBM MQ , spusťte danou službu a otestujte klienta.

Implementace klienta webové služby do prostředí Axis2 pro použití transportu produktu IBM MQ pro protokol SOAP
Připravte adresář implementace a konfigurační soubor Axis2 pro klienta. Zadejte proxy klienta a třídu klienta a nastavte proměnnou CLASSPATH. Nakonfigurujte fronty a kanály produktu IBM MQ , spusťte danou službu a otestujte klienta.

Implementace na klienta Axis2 pomocí protokolu W3C SOAP prostřednictvím produktu JMS
Webová služba, která je svázána s doporučujícím doporučením W3C pro protokol SOAP over JMS , musí být spuštěna v kontejneru EJB aplikačního serveru Java EE. Tato úloha je krokem 4 připojení klienta webové služby Axis2 a webové služby implementované do produktu WebSphere Application Server pomocí protokolu W3C protokolu SOAP prostřednictvím protokolu JMS . Upravte adresu URL v klientovi Axis2 vyvinutém pro transport IBM MQ pro protokol SOAP tak, aby bylo možné použít doporučení kandidáta W3C pro protokol SOAP prostřednictvím produktu JMS.

Připojte klienta Axis2 ke službě JAX-WS pomocí protokolu SOAP W3C nad JMS a WebSphere Application Server .

Po dokončení této úlohy se bude volat webová služba JAX-WS, která je spuštěna v produktu WebSphere Application Server , z klienta Axis2 . Klient Axis2 a WebSphere Application Server používají doporučení kandidáta W3C pro protokol SOAP přes JMS spuštěný na serveru IBM MQ. Pomocí Eclipse Galileo a Rational Application Developer můžete sestavit klienta webové služby a webovou službu.

Než začnete

Úloha vyžaduje verzi 7 produktu Rational Software Development Environment a WebSphere Application Server. Úloha byla vytvořena pomocí produktu Rational Application Developer zabaleného s produktem Rational Software Architect for WebSphere Software 7.5.5.1a WebSphere Application Server 7.0 Test Environment 7.0.0.9 Update 1. Požadujete také produkt IBM MQ 7.0.1.3nebo novější.

Úloha je založena na dvou dalších úlohách, profilu Liberty profilea “Vyvíjení klienta JAX-WS pro produkt WebSphere transport pro protokol SOAP s použitím prostředí Eclipse” na stránce 1285. Chcete-li dokončit tyto úlohy, vaše vývojové prostředí již má Eclipse Galileo, profil Liberty Profile, modul plug-in Eclipse pro Liberty a Axis2 1.4.1 .

Některé kroky jsou složité. Tento postup předpokládá, že při vyvíjení aplikací webových služeb pro produkt WebSphere Application Server s použitím produktu Rational Application Developer. Požadavky na procesor a paměť pro úlohu jsou velké. Úloha byla provedena ve virtuálním počítači VMWare Windows 7 SP1 alokoval 1.8GB paměti.

Nainstalujte všechen software před spuštěním úlohy. Software trvá asi den stažení a jeden den k instalaci, v závislosti na šířce pásma. Úloha trvá alespoň půl dne.

Informace o této úloze

Scénář této úlohy spočívá v tom, že jste vyvinuli webovou službu s cenovou nabídkou, `StockQuoteAxiss` použitím otevřeného zdrojového nástroje Eclipse Galileo. Produkt `StockQuoteAxis` je implementován pomocí protokolu SOAP prostřednictvím protokolu HTTP, který je spuštěn na otevřeném zdrojovém serveru, profilu Liberty.

Chcete svázat webové služby, které implementujete na přenos systému zpráv založené na standardu, například SOAP prostřednictvím produktu JMS, nebo na spolehlivý systém zpráv webových služeb a také na protokol SOAP prostřednictvím protokolu HTTP. Chcete, aby klient i služba používaly standardní rozhraní založená na rozhraní. Z tohoto důvodu, i když váš budoucí projektový tým realizoval řešení pomocí přenosu IBM MQ pro SOAP, jste nešel do výroby.

Klient Axis2 odstranil problém, že klient SOAP pro přenos IBM MQ pro protokol SOAP vyžaduje změnu od klienta HTTP. Problém zůstává i nadále, že služba připojená přenosem IBM MQ pro protokol SOAP je hostována speciálním modulem listener poskytovaným produktem IBM MQ: `SimpleJavaListener`.

Při použití standardu W3C SOAP přes JMS ve stavu doporučeného doporučení někteří dodavatelé poskytují podporu pro SOAP W3C přes JMS. Podpora vám umožňuje implementovat webovou službu na aplikační server a připojit se ke stejné službě pomocí různých protokolů konektivity. Podpora poskytovaná produktem WebSphere Application Server v7 odstraňuje problém s tím, že bude hostovat webovou službu samostatně, aby bylo možné použít přenos SOAP založený na zprávě. Použití rozhraní pro přenos zpráv založené na standardu JMSznamená, že můžete vyvíjet řešení pomocí nástrojů různých dodavatelů. Doufáme, že nástroje webových služeb v produktu Eclipse budou v budoucnu zahrnovat vazby SOAP přes JMS .

Většina kroků se provádí pomocí platformy Eclipsenebo pomocí nástrojů správy dodávaných s produkty WebSphere . Kroky jsou popsány pro prostředí produktu Windows . S drobnými úpravami některých příkazů můžete provádět tyto kroky na jiných platformách.

Vypíší se přípravné kroky pro vytvoření webové služby HTTP a připojení k ní pomocí prostředí Axis2 . Klient a WSDL z těchto kroků se používají k vytvoření řešení.

Postup

1. Připojte se k webové službě Axis StockQuotepomocí klienta Axis2 a přenosu IBM MQ pro SOAP
   a) Použijte profil Liberty
   b) “Vyvíjení klienta JAX-WS pro produkt WebSphere transport pro protokol SOAP s použitím prostředí Eclipse” na stránce 1285
   c) “Implementace klienta webové služby do prostředí Axis2 pro použití transportu produktu IBM MQ pro protokol SOAP” na stránce 1313

2. Připojte se k webové službě osy StockQuotepomocí klienta Axis2 a doporučeného doporučení W3C pro protokol SOAP prostřednictvím produktu JMS.
   a) Nakonfigurujte prostředky produktu IBM MQ .
   b) Nakonfigurujte prostředky produktu WebSphere Application Server .
   c) “Vývoj webové služby JAX-WS pro W3C SOAP přes JMS” na stránce 1272
   d) “Implementace na klienta Axis2 pomocí protokolu W3C SOAP prostřednictvím produktu JMS” na stránce 1316

Poznámky

Tyto informace byly vyvinuty pro produkty a služby poskytované v USA.

Společnost IBM nemusí nabízet produkty, služby nebo funkce uvedené v tomto dokumentu v jiných zemích. Informace o produktech a službách, které jsou ve vaší oblasti aktuálně dostupné, získáte od místního zástupce společnosti IBM. Odkazy na produkty, programy nebo služby společnosti IBM v této publikaci nejsou míněny jako vyjádření nutnosti použití pouze uvedených produktů, programů či služeb společnosti IBM. Místo toho lze použít jakýkoli funkčně ekvivalentní produkt, program nebo službu, které neporušují žádná práva k duševnímu vlastnictví IBM. Ověření funkčnosti produktu, programu nebo služby pocházející od jiného výrobce je však povinností uživatele.

Společnost IBM může vlastnit patenty nebo nevyřízené žádosti o patenty zahrnující předměty popsané v tomto dokumentu. Vlastnictví tohoto dokumentu neposkytuje licenci k těmto patentům. Dotazy týkající se licencí můžete posílat písemně na adresu:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Odpovědi na dotazy týkající se licencí pro dvoubajtové znakové sady (DBCS) získáte od oddělení IBM Intellectual Property Department ve vaší zemi, nebo tyto dotazy můžete zasílat písemně na adresu:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

Následující odstavec se netýká Velké Británie nebo kterékoliv jiné země, kde taková opatření odporují místním zákonům: SPOLEČNOST INTERNATIONAL BUSINESS MACHINES CORPORATION TUTO PUBLIKACI POSKYTUJE TAKOVOU, "JAKÁ JE" , BEZ JAKÝCHKOLIV ZÁRUK, VYJÁDŘENÝCH VÝSLOVNĚ NEBO VYPLÝVAJÍCÍCH Z OKOLNOSTÍ, VČETNĚ, A TO ZEJMÉNA, ZÁRUK NEPORUŠENÍ PRÁV TŘETÍCH STRAN, PRODEJNOSTI NEBO VHODNOSTI PRO URČITÝ ÚČEL VYPLÝVAJÍCÍCH Z OKOLNOSTÍ. Některé právní řády u určitých transakcí nepřipouštějí vyloučení záruk výslovně vyjádřených nebo vyplývajících z okolností, a proto se na vás toto omezení nemusí vztahovat.

Uvedené údaje mohou obsahovat technické nepřesnosti nebo typografické chyby. Údaje zde uvedené jsou pravidelně upravovány a tyto změny budou zahrnuty v nových vydáních této publikace. Společnost IBM může kdykoli bez upozornění provádět vylepšení nebo změny v produktech či programech popsaných v této publikaci.

Veškeré uvedené odkazy na webové stránky, které nespravuje společnost IBM, jsou uváděny pouze pro referenci a v žádném případě neslouží jako záruka funkčnosti těchto webů. Materiály uvedené na tomto webu nejsou součástí materiálů pro tento produkt IBM a použití uvedených stránek je pouze na vlastní nebezpečí.

Společnost IBM může použít nebo distribuovat jakékoli informace, které jí sdělíte, libovolným způsobem, který společnost považuje za odpovídající, bez vyžádání vašeho svolení.

Vlastníci licence k tomuto programu, kteří chtějí získat informace o možnostech (i) výměny informací s nezávisle vytvořenými programy a jinými programy (včetně tohoto) a (ii) oboustranného využití vyměňovaných informací, mohou kontaktovat informační středisko na adrese:

IBM Corporation
Koordinátor spolupráce softwaru, oddělení 49XA
148 00 Praha 4-Chodby

148 00 Praha 4-Chodov
U.S.A.

Poskytnutí takových informací může být podmíněno dodržením určitých podmínek a požadavků zahrnujících v některých případech uhrazení stanoveného poplatku.

IBM poskytuje licencovaný program popsaný v těchto informacích a veškeré dostupné licencované materiály na základě podmínek smlouvy IBM Customer Agreement, IBM International Program License Agreement nebo jiné ekvivalentní smlouvy mezi námi.

Jakékoli údaje o výkonnosti obsažené v této publikaci byly zjištěny v řízeném prostředí. Výsledky získané v jakémkoli jiném operačním prostředí se proto mohou výrazně lišit. Některá měření mohla být prováděna na vývojových verzích systémů a není zaručeno, že tato měření budou stejná i na běžně dostupných systémech. Některá měření mohla být navíc odhadnuta pomocí extrapolace. Skutečné výsledky mohou být jiné. Čtenáři tohoto dokumentu by měli zjistit použitelné údaje pro své specifické prostředí.

Informace týkající se produktů jiných výrobců pocházejí od dodavatelů těchto produktů, z jejich veřejných oznámení nebo z jiných veřejně dostupných zdrojů. Společnost IBM tyto produkty netestovala a nemůže potvrdit správný výkon, kompatibilitu ani žádné jiné výroky týkající se produktů jiných výrobců než IBM. Otázky týkající se kompatibility produktů jiných výrobců by měly být směrovány dodavatelům těchto produktů.

Veškerá tvrzení týkající se budoucího směru vývoje nebo záměrů společnosti IBM se mohou bez upozornění změnit nebo mohou být zrušena a reprezentují pouze cíle a plány společnosti.

Tyto údaje obsahují příklady dat a sestav používaných v běžných obchodních operacích. Aby byla představa úplná, používají se v příkladech jména osob a názvy společností, značek a produktů. Všechna tato jména a názvy jsou fiktivní a jejich podobnost se jmény, názvy a adresami používanými ve skutečnosti je zcela náhodná.

LICENČNÍ INFORMACE:

Tyto informace obsahují ukázkové aplikační programy ve zdrojovém jazyce ilustrující programovací techniky na různých operačních platformách. Tyto ukázkové programy můžete bez závazků vůči společnosti IBM jakýmkoli způsobem kopírovat, měnit a distribuovat za účelem vývoje, používání, odbytu či distribuce aplikačních programů odpovídajících rozhraní API pro operační platformu, pro kterou byly ukázkové programy napsány. Tyto příklady nebyly plně testovány za všech podmínek. Společnost IBM proto nemůže zaručit spolehlivost, upotřebitelnost nebo funkčnost těchto programů.

Při prohlížení těchto dokumentů v elektronické podobě se nemusí zobrazit všechny fotografie a barevné ilustrace.

Informace o programovacím rozhraní

Informace programátorských rozhraní, je-li poskytnuta, vám pomohou vytvořit aplikační software pro použití s tímto programem.

Tato příručka obsahuje informace o zamýšlených programovacích rozhraních, které umožňují zákazníkům psát programy za účelem získání služeb produktu WebSphere MQ.

Tyto informace však mohou obsahovat i diagnostické údaje a informace o úpravách a ladění. Informace o diagnostice, úpravách a vyladění jsou poskytovány jako podpora ladění softwarových aplikací.

Důležité: Nepoužívejte tyto informace o diagnostice, úpravách a ladění jako programátorské rozhraní, protože se mohou měnit.

Ochranné známky

IBM, logo IBM , ibm.comjsou ochranné známky společnosti IBM Corporation, registrované v mnoha jurisdikcích po celém světě. Aktuální seznam ochranných známek IBM je k dispozici na webu na stránce "Copyright and trademark information"www.ibm.com/legal/copytrade.shtml. Ostatní názvy produktů a služeb mohou být ochrannými známkami společnosti IBM nebo jiných společností.

Microsoft a Windows jsou ochranné známky společnosti Microsoft Corporation ve Spojených státech a případně v dalších jiných zemích.

UNIX je registrovaná ochranná známka skupiny The Open Group ve Spojených státech a případně v dalších jiných zemích.

Linux je registrovaná ochranná známka Linuse Torvaldse ve Spojených státech a případně v dalších jiných zemích.

Tento produkt obsahuje software vyvinutý v rámci projektu Eclipse Project (http://www.eclipse.org/).

Java a všechny ochranné známky a loga založené na termínu Java jsou ochranné známky nebo registrované ochranné známky společnosti Oracle anebo příbuzných společností.

Číslo položky:

(1P) P/N: